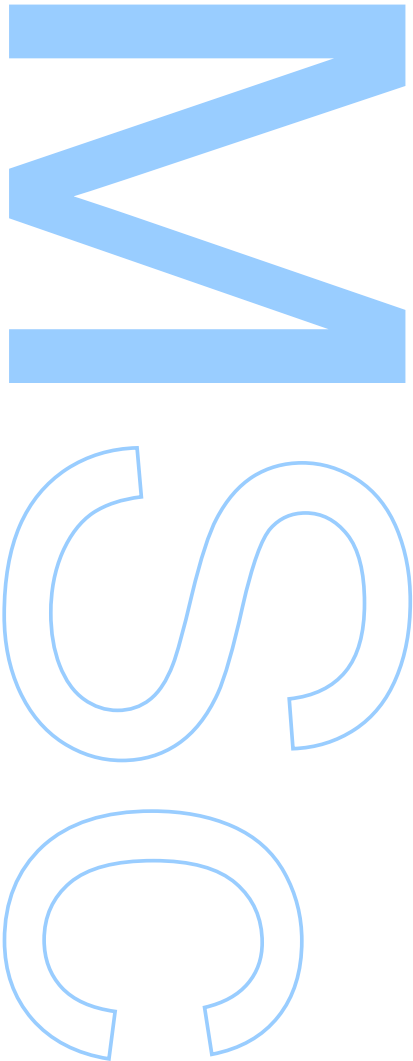
View metadata, citation and similar pape

Finite Transducers in Public Key Cryptography

Joana Barão Vieira
Master's Degree in Information Security
Departamento de Ciência de Computadores
2017

Orientador
Rogério Ventura Lages dos Santos Reis, Professor Auxiliar,
Faculdade de Ciências da Universidade do Porto

Coorientador
Ivone de Fátima da Cruz Amorim, ,
Centro de Matemática da Universidade do Porto



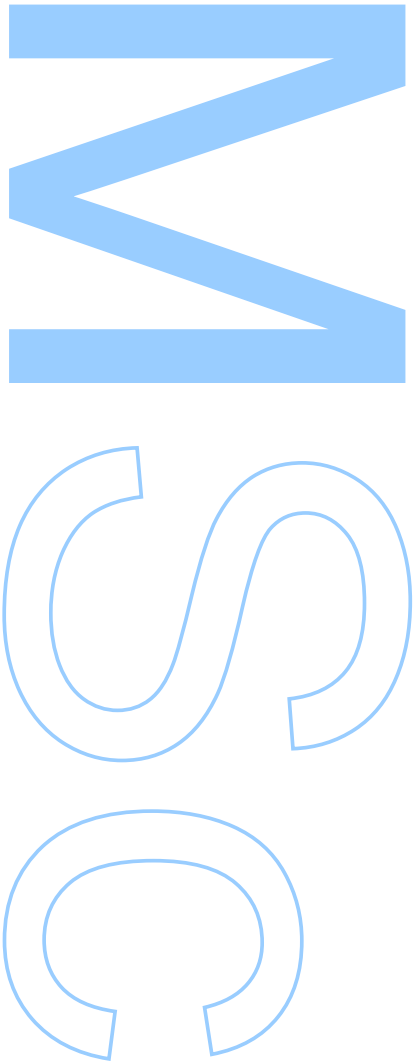
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____



Dedicated to my parents, for all the money spent on my education

Abstract

Cryptography has been used for secure communication for thousands of years. However, today, cryptography faces new challenges. The increase of computing power, the increasing use of small devices like smart cards, as well as the possibility of quantum computing becoming a reality, require new cryptographic systems that offer security relying on different assumptions of classic ones. Furthermore, it is necessary small key sizes as well as fast encryption and decryption.

Finite Automata Public Key Cryptosystems, FAPKCs, are systems based on finite transducers, that meet the previous requirements. Their security does not rely on complexity assumptions related to number theory problems, as classical ones. Also, these cryptosystems offer relatively small key sizes as well as linear encryption and decryption times complexity.

As in other cryptosystems, a fundamental concept in FAPKC is the capacity of inverting the encryption procedure, in a way that is hard to anyone except to the owner of the private key. In this sense, these cryptosystems depend heavily on results of invertibility of finite transducers, as well as, results about the special product used to generate the public key. In this dissertation, we study this two concepts and present a wide variety of examples in order to help the readers to understand them.

The main contributions of this work are the extended definition of quasi-linear finite transducers with memory, the formalization of a procedure to check injectivity and construct inverses of finite transducers with memory (linear and quasi-linear) and the extension to the Bao-Igarashi attack to FAPKC to all possible values of injectivity delay.

Keywords: finite transducers, invertibility, composition, attack to FAPKC.

Resumo

A criptografia tem sido usada para comunicações seguras durante milhares de anos. No entanto, hoje, a criptografia enfrenta novos desafios. O aumento do poder computacional, o uso crescente de pequenos dispositivos como smart cards, assim como a possibilidade de a computação quântica se tornar realidade, requerem novos sistemas criptográficos que ofereçam segurança baseada em pressupostos diferentes dos clássicos. Além disso, é necessário pequenos tamanhos de chave, bem como processos de cifrar e decifrar rápidos.

Os Finite Automata Public Key Cryptosystems, FAPKCs, são sistemas baseados em transdutores finitos, que preenchem os requisitos anteriores. A segurança destes sistemas não depende de pressupostos de complexidade relacionados com problemas de teoria dos números, como os clássicos. Além disso, os sistemas FAPKC oferecem tamanhos de chave relativamente pequenos, assim como tempo linear a cifrar e decifrar.

Como em outros sistemas criptográficos, um conceito fundamental nos FAPKCs é a capacidade de inverter a cifra, de forma a que seja difícil para qualquer pessoa exceto para o proprietário da chave privada. Neste sentido, estes sistemas dependem fortemente dos resultados relacionados com a invertibilidade de transdutores finitos e, para além disso, dos resultados relacionados com o produto especial usado para gerar a chave pública. Nesta dissertação, estudamos estes conceitos e apresentamos uma grande variedade de exemplos para ajudar os leitores a compreendê-los.

As principais contribuições deste trabalho são a definição estendida de transdutores finitos quase lineares com memória, a formalização de um procedimento para verificar a injetividade e construir inversos de transdutores finitos com memória (lineares e quase lineares) e a extensão do Bao-Igarashi ataque ao FAPKC para todos os valores possíveis para o atraso da injetividade.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography has been used for secure communication for thousands of years. Throughout history, military communication has had the greatest influence on cryptography and its advancements. The need for secure commercial and private communication has been led by the Information Age. Until the invention of public key cryptography, all ciphers were symmetric. Symmetric cryptography uses algorithms that have a key to encrypt and decrypt information. This means that each party to the communication need the same key, the sender to encrypt the message, and the recipient to decrypt it. This presented a significant problem: before one could communicate securely, it was needed to exchange a secret with the partner. Public key cryptosystems hugely revolutionized cryptography by dramatically simplifying this key distribution process. Rather than sharing secret keys, users could now transmit their public key to other parties. This public key allowed the sender to encrypt, but it could not be used to perform the corresponding decryption operation. That part would be done with the corresponding private key, kept as secret by the recipient.

The concept of public key cryptography was introduced by Diffie, Hellman and Merkle in 1976. In 1978, Rivest, Shamir, and Adleman presented the first public key cryptosystem, called RSA [Dif88]. Its security is connected to the difficulty of factoring large integers. The ElGamal cryptosystem, invented by Taher ElGamal in 1985 [ElG85], relies on a similar problem, the discrete logarithm problem. Also in 1985, Neal Koblitz [Kob87] and Victor Miller [Mil85], independently, introduced the elliptic curve cryptography, based on the elliptic curve discrete logarithm problem. Although mathematically more complex, elliptic curves provide smaller key

sizes and faster operations for approximately equivalent estimated security. Since the 1970s, a large variety of public key cryptosystems have been developed, most of them based on complexity assumptions related to the same number theory problems as RSA and ElGamal. This dependence on a very small set of problems makes such cryptosystems somewhat vulnerable.

In a series of papers [TCC97, TC97, TC99], Renji Tao introduced a family of cryptosystems based on finite transducers, named FAPKCs (which stands for Finite Automata Public Key Cryptosystems). The security of these cryptosystems does not rely in complexity assumptions related to number theory problems, rather relying on the difficulty of inverting non-linear finite transducers and of factoring matrix polynomials over $\mathbb{F}_q$ [Tao09], both NP-problems. Futhermore, this family of cryptosystems uses relatively small key sizes, fast encryption and decryption and can also be used to signature. The implementation of FAPKC only require logical operations, making it suitable for smart card applications [TC97].

In the FAPKCs, roughly speaking, the private key consists of two finite transducers with memory, one linear and one quasi-linear. The public key is obtained by a special product of the original pair, resulting in a non-linear finite transducer with memory. It is not known an algorithm to invert non-linear finite transducers, neither to factorize them. Therefore, in order to invert the public key transducer, one needs the inverses of its factors, which are easily computed from the transducers in the private key. The main difference between the different variants of FAPKC is the choice of the type of transducers for the private key.

Although some of FAPKC schemes were already proved to be insecure [BI95], these cryptosystems continue to present as a good alternative to the classic ones. Despite the many advantages of FAPKC, its study was somehow condemned by the fact that many papers were written in an arid language, with many results presented without proof or examples, and others referring Chinese papers. Some clarification and consolidation of the work already done on this subject were presented by Ivone Amorim, António Machiavelo and Rogério Reis in a serie of papers [AMR12, AMR14a, AMR14b, AMR14c] and a PhD thesis [dCA16]. In these works, it was presented in a more clear way some results already known, it was studied the number and size of equivalence classes of linear finite transducers defined over $\mathbb{F}_q$, as well as algorithms to check invertibility and invert linear transducers with memory. This dissertation is the continuation of that work.

In this work, after presenting some known results about linear finite transducers and finite transducers with memory, we introduce a new "extended" definition of quasi-linear finite transducers with memory, allowing the increase of possible private keys. Then, we introduce and formalize a procedure to check injectivity of linear and quasi-linear finite transducers, as well as a necessary and sufficient condition for the injectivity of these transducers. Inverting linear and quasi-linear transducers with memory is fundamental in the key generation process of FAPKCs, since one needs to define both an invertible transducer with memory and a corresponding inverse. In order to be able to study the preexisting FAPKCs, we present two types of compositions of finite transducers with memory, introduced by Tao [Tao09]. Also, it will be illustrated a general scheme of key generation, and encryption and decryption processes. The scheme presented is known to be insecure, however, it is the only one that we can understand through the papers we had access. Finally, we will present the Bao-Igarashi attack to this scheme and extend it to work with all possible values of transducers injectivity delay. Throughout this work, it is present a wide variety of examples to illustrate the concepts and procedures introduced.

1.1 Structure of this Dissertation

We start by reviewing, in Chapter 2, several concepts and results from different areas of mathematics that will be used throughout this work. We also introduce some convenient notation.

Preliminary notions and results of general finite transducers are given in Chapter 3, including the concepts of injectivity and invertibility that are considered in this work. Also, in this chapter, we give the definitions of linear finite transducer and finite transducer with memory (linear and quasi-linear). Then, we present our new extended definition of quasi-linear finite transducers with memory. Some results about invertibility of finite transducers with memory are also presented in this chapter.

In Chapter 4, it is given a necessary and sufficient condition to invertibility of linear finite transducers with memory. We present and formalize a procedure to check injectivity of linear finite transducers with memory, using R_a and R_b transformations. During this procedure, we construct an inverse transducer of the original one. Also, these results are extended to quasi-linear finite transducers with memory.

Chapter 5 is devoted to the presentation of Bao-Igarashi attack to FAPKC. In order to be able to do that, we start by introducing two different compositions of finite transducers. Next, we present a general description of FAPKC as well as a base scheme with key generation and encryption and decryption processes. Lastly, it is presented the Bao-Igarashi attack modified to work with all possible values of injectivity delay and illustrate it through an example.

To end, in Chapter 6, we summarise our contributions and discuss some future research directions.

Chapter 2

Mathematical Prerequisites

2.1 Relations and Funtions

Let A and B be two sets. A *relation* $\sim$ from A to B is a subset of the cartesian product $A \times B$. The notation $a \sim b$ is used to denote that (a, b) is in the relation $\sim$. If (a, b) is not in the relation $\sim$, it is denoted $a \not\sim b$. When $A = B$, $\sim$ is also called a *binary relation on* A.

A binary relation $\sim$ on a set A is said to be an *equivalence relation* if and only if the following conditions hold:

- $\sim$ is reflexive, i.e., $a \sim a$, for all a in A;
- $\sim$ is symmetric, i.e., $a \sim b$ if and only if $b \sim a$, for all a, b in A;
- $\sim$ is transitive, i.e., if $a \sim b$ and $b \sim c$, then $a \sim c$, for all a, b, c in A.

Let $\sim$ be an equivalence relation on A. For any $a \in A$, the set $[a]_\sim = \{b \in A \mid a \sim b\}$ is called the *equivalence class* containing a, while the set of all equivalence classes, $A/\sim = \{[a]_\sim \mid a \in A\}$, is called the *quotient of* A *by* $\sim$.

The *restriction of a binary relation* on a set A to a subset S is the set of all pairs (a, b) in the relation for which a and b are in S. If a relation is an equivalence relation, its restrictions are too.

Given a positive integer n, an example of an equivalence relation is the *congruence modulo* n

relation on the set of integers, $\mathbb{Z}$. For a positive integer n, one defines this relation on $\mathbb{Z}$ as follows. Two integers a and b are said to be *congruent modulo* n, written:

$$a \equiv_n b \quad \text{or} \quad a \equiv b \pmod{n},$$

if their difference $a - b$ is a multiple of n. It is easy to verify that this is an equivalence relation on the integers. The number n is called the *modulus*. An equivalence class consists of those integers which have the same remainder on division by n. The set of *integers modulo* n, which is denoted by $\mathbb{Z}_n$, is the set of all congruence classes of the integers for the modulus n.

Example 2.1.1. *Take $n = 2$. Then, for example,*

$$5 \equiv 3 \equiv 1 \pmod{2} \quad \textit{and} \quad [1]_\sim = \{2j + 1 \mid j \in \mathbb{Z}\}.$$

A relation from a set A to a set B is called a *function*, *map* or *mapping*, if each element of A is related to exactly one element in B. A function f from A to B is denoted by $f : A \to B$, and for all a in A, $f(a)$ denotes the element in B which is related to a, which is usually called the *image of a under f*.

A function $f : A \to B$ is called *injective*, or a *one-to-one* function, if it satisfies the following condition:

$$\forall\, a, a' \in A,\ f(a) = f(a') \Rightarrow a = a',$$

and is called *surjective* if the following condition holds:

$$\forall\, b \in B,\ \exists\, a \in A,\ f(a) = b.$$

If a function is both injective and surjective, then it is called *bijective* or a *bijection*.

2.2 Groups, Rings and Fields

Let A be a set and n a natural number. A *n-ary operation* on A is a mapping from A^n to A. The operation $\diamond : A^2 \to A$ is called a *binary operation*, which only means that if (a, b) is an ordered pair of elements of A, then $a \diamond b$ is a unique element of A.

A *group* is an ordered pair $(G, \diamond)$, where G is a non-empty set and $\diamond$ is a binary operation on G (called the *group operation*), satisfying the following properties:

- the operation $\diamond$ is associative, that is, $x \diamond (y \diamond z) = (x \diamond y) \diamond z$, for all $x, y, z \in G$;
- there is an element $e \in G$ such that $x \diamond e = e \diamond x = x$, for all x in G. Such an element is unique and is called the *identity element*;
- if x is in G, then there is an element y in G such that $x \diamond y = y \diamond x = e$, where e is the identity element. That element y is called the *inverse* of x.

A group is *denoted additively* (*multiplicatively*) or is an *additive* (*multiplicative*) *group* when:

- the group operation is denoted by $+$ $(\cdot)$;
- the identity element is denoted by 0 (1);
- the inverse of an element x is denoted by $-x$ (x^{-1}),

respectively. If the group operation is commutative, i.e., $x \diamond y = y \diamond x$ for all x, y in G, then G is called an *Abelian group* or *commutative group*.

There are some very familiar examples of Abelian groups under addition, namely the integers $\mathbb{Z}$, the rationals $\mathbb{Q}$, the real numbers $\mathbb{R}$, and $\mathbb{Z}_n$, for $n \in \mathbb{N}$. Notice that $\mathbb{N}$ denotes de set of natural numbers, i.e., $\mathbb{N} = \{1, 2, 3, \ldots\}$.

A *ring* is an ordered triple $(R, +, \cdot)$, where R is a non-empty set, $+$ is a binary operation on R called *addition*, and $\cdot$ is also a binary operation on R called *multiplication*, which obey the following rules:

- $(R, +)$ is an Abelian group (the additive identity is denoted by 0);
- the multiplicative operation is associative, that is, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$, for all x, y, z in R;
- there is an element 1 in R such that $1 \cdot x = x \cdot 1 = x$, for all x in R. 1 is called the *multiplicative identity*;
- the multiplication is left distributive with respect to addition, that is, $x \cdot (y + z) = x \cdot y + x \cdot z$, for all x, y, z in R;
- the multiplication is right distributive with respect to addition, i.e., $(x + y) \cdot z = x \cdot z + y \cdot z$, for all x, y, z in R.

A simple example of a ring is the set of integers with the usual operations of addition and multiplication.

Let R be a ring with multiplicative identity 1. An element r in R is said to be *multiplicatively invertible* or just *invertible* if and only if there is an element s in R such that $r \cdot s = s \cdot r = 1$, and s is called the *multiplicative inverse* of r or just the *inverse* of r. An invertible element in R is called an *unit* and the set of units of R is represented by R^*. Let $a, b \in R$. We say that *a divides b*, and write $a \mid b$, if there is $q \in R$ such that $b = aq$, where aq abbreviates $a \cdot q$. The definition of congruence modulo n relation on the set of integers, presented before, can be generalised to elements of a ring. Thus, two elements a, b in a ring, R, are *congruent modulo* $n \in R$ if $n \mid (a - b)$.

A *field* is a commutative ring that has multiplicative inverses for all non-zero elements. The set of real numbers, together with the usual operations of addition and multiplication, is a field.

If $\mathbb{F}$ is a field with a finite number of elements, then one says that $\mathbb{F}$ is a *finite field* or a *Galois field*. The simplest examples of finite fields are the prime fields: given a prime number p, the prime field $\mathbb{F}_p$ or $GF(p)$ is the set of integers modulo p, previously denoted by $\mathbb{Z}_p$. The elements of a prime field may be represented by integers in the range $0, 1, \ldots, p-1$. For example, $\mathbb{F}_2 = \{0, 1\}$.

2.3 Modules and Vector Spaces

Let R be a ring and 1 its multiplicative identity. A *right R-module*, M, consists of an Abelian group $(M, +)$ and an operation $* : M \times R \to M$ such that, for all $r, s \in R$ and $x, y \in M$:

- $(x + y) * r = x * r + y * r$
- $x * (r + s) = x * r + x * s$
- $x * (rs) = (x * r) * s$
- $x * 1 = x$.

The operation of the ring on M is called *scalar multiplication* and is usually written by juxtaposition, i.e., xr for $r \in R$ and $x \in M$. However, in the definition above, it is denoted as $x * r$

to distinguish it from the ring multiplication operation, which is denoted by juxtaposition. A *left R-module* M is defined similarly, except that the ring acts on the left, i.e., scalar multiplication takes the form $* : R \times M \to M$ and the above axioms are written with scalars r and s on the left of x and y.

If R is commutative, then left R-modules are the same as right R-modules and are simply called *R-modules*.

For example, if R is a commutative ring and $n \in \mathbb{N}$, then R^n is both a left and a right R-module if one uses the component-wise operations:

$$(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n),$$
$$\alpha(a_1, a_2, \ldots, a_n) = (\alpha a_1, \alpha a_2, \ldots, \alpha a_n),$$

for all $(a_1, a_2, \ldots, a_n)$, $(b_1, b_2, \ldots, b_n) \in R^n$, and for all $\alpha \in R$.

Let $\mathbb{F}$ be a field. Then an $\mathbb{F}$-module is called a *vector space* over $\mathbb{F}$.

Example 2.3.1. *Let $n \in \mathbb{N}$. The set $\mathbb{F}_2^n$ with the component-wise operations of addition and scalar multiplication, as defined above, is a vector space over the field $\mathbb{F}_2$ which is denoted simply by $\mathbb{F}_2^n$.*

Let V be a vector space over a field $\mathbb{F}$. A non-empty subset U of V is said to be a *subspace* of V if U is itself a vector space over $\mathbb{F}$ with the same operations as V.

Let V be a vector space over an arbitrary field $\mathbb{F}$ and $n \in \mathbb{N}$. A vector of the form

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n,$$

where $\alpha_i \in \mathbb{F}$ and $v_i \in V$, for $i = 1, \ldots, n$, is called a *linear combination* of the vectors $v_1, v_2, \ldots, v_n$. The scalar α_i is called the *coefficient* of v_i, for $i = 1, \ldots, n$.

The set of all linear combinations of given vectors $v_1, v_2, \ldots, v_n \in V$ is a subspace of V and is called the *subspace generated by* (or *spanned by*) the vectors $v_1, v_2, \ldots, v_n$.

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a non-empty subset of V and $v \in V$. If there are scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ in $\mathbb{F}$ such that

$$v = \alpha_1 s_1 + \alpha_2 s_2 + \ldots + \alpha_m s_n,$$

then one says that v can be written as a *linear combination* of the vectors in S. The set S is *linearly independent* if and only if no vector in S can be written as a linear combination of the other vectors in that set. If one vector in S can be written as a linear combination of the others, then the set of vectors is said to be *linearly dependent*.

A non-empty subset B of V is said to be a *basis* of V if and only if both of the following are true:

- B is a linearly independent set;
- V is spanned by B.

Example 2.3.2. *It is easy to see that the set* $\{(1,0,0);(0,1,0);(0,0,1)\}$ *is a basis of* $\mathbb{R}^3$*, which is called the* standard basis *of* $\mathbb{R}^3$*.*

If V is a vector space that has a basis B containing a finite number of vectors, then V is said to be *finite dimensional*. The number of elements on that basis is what is called the *dimension* of V, and is denoted by $\dim(V)$. It can be shown that the dimension of a vector space does not depend on the basis chosen, since all the bases have the same number of elements [Val93]. If V has no finite basis, then V is said to be *infinite dimensional*.

Example 2.3.3. *From the previous example, it is clear that* $\mathbb{R}^3$ *is finite dimensional and its dimension is* 3*.*

2.4 Matrices and Smith Normal Form

Let $m, n \in \mathbb{N}$ and R a field. Let $a_{i,j} \in R$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The rectangular array A defined by

$$A = [a_{i,j}] = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

is called a *matrix over* R with m rows and n columns, or simply an $m \times n$ *matrix*. If $m = n$ one says that A is a *square matrix*. If $m \neq n$, then the matrix is said to be *non-square*. The set of all

matrices over R with m rows and n columns is denoted by $\mathcal{M}_{m\times n}(R)$. If $m = n$, one denotes $\mathcal{M}_{n\times n}(R)$ simply by $\mathcal{M}_n(R)$. The elements of a matrix are called its *entries*, and $a_{i,j}$ denotes the entry that occurs at the intersection of the ith row and jth column.

A matrix in $\mathcal{M}_{m\times n}(R)$ ($\mathcal{M}_n(R)$) in which each element is the additive identity of R is called a *zero matrix*, or *null matrix*, and is usually denoted by $0_{m\times n}$ (0_n).

Example 2.4.1. *The null matrices in $\mathcal{M}_3(R)$ and $\mathcal{M}_{2\times 4}(R)$ are, respectively,*

$$0_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \textit{and} \quad 0_{2\times 4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The $n \times n$ matrix $A = [a_{i,j}]$ over R such that $a_{i,i} = 1$ and $a_{i,j} = 0$, for $i \neq j$, is called the *identity matrix* of order n over R and is denoted by I_n.

Example 2.4.2. *The identity matrix of order 2 is* $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

A $m \times n$ matrix $A = [a_{i,j}]$ can be thought of either as a collection of m row vectors, each having n coordinates:

$$\begin{matrix} [a_{1,1} & a_{1,2} & \dots & a_{1,n}], \\ [a_{2,1} & a_{2,2} & \dots & a_{2,n}], \\ & & \vdots & \\ [a_{m,1} & a_{m,2} & \dots & a_{m,n}], \end{matrix}$$

or as a collection of n column vectors, each having m coordinates:

$$\begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{m,1} \end{bmatrix}, \begin{bmatrix} a_{1,2} \\ a_{2,2} \\ \vdots \\ a_{m,2} \end{bmatrix}, \dots, \begin{bmatrix} a_{1,n} \\ a_{2,n} \\ \vdots \\ a_{m,n} \end{bmatrix}.$$

The subspace of R^n generated by the row vectors of A is called the *row space* of the matrix A. The dimension of this row space is called the *row rank* of A. Similarly, the subspace of R^m generated by the column vectors of A is called the *column space* of A, and its dimension is the *column rank* of A.

It is well known that the row rank of a matrix is equal to its column rank [McC71]. Therefore, one does not need to distinguish between the row rank and the column rank of a matrix. The common value of the row rank and the column rank of a matrix is called simply the *rank* of the matrix. The rank of a matrix A is here denoted by $rank(A)$.

A matrix is said to have *maximal rank* if its rank equals the lesser of the number of rows and columns.

Example 2.4.3. *Consider the matrices*

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \textit{and} \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

defined over $\mathbb{F}_2$*. Then, since* $rank(A) = 2 =$ *number of rows,* A *has maximal rank. The matrix* B *does not have maximal rank because* $rank(B) = 1 <$ *number of rows* $<$ *number of columns.*

One can define two operations that give $\mathcal{M}_n(R)$ a ring structure. Let $A = [a_{i,j}]$ and $B = [b_{i,j}]$ be matrices in $\mathcal{M}_{m\times n}(R)$. The sum of A and B is the $m \times n$ matrix $C = [c_{i,j}]$ such that

$$c_{i,j} = a_{i,j} + b_{i,j}.$$

Now, let $A = [a_{i,j}]$ be a matrix in $\mathcal{M}_{m\times n}(R)$ and $B = [b_{i,j}]$ a matrix in $\mathcal{M}_{n\times p}(R)$. The matrix product $C = [c_{i,j}] = AB$ is the $m \times p$ matrix defined by

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

The set $\mathcal{M}_n(R)$ together with the two operations defined above is a ring, which is not commutative. Notice that the addition of matrices is defined only for matrices of the same size, and the product is defined between matrices such that the number of columns of the first matrix equals the number of rows of the second one.

Example 2.4.4. *Consider the matrices* A *and* B *of the previous example. Then*

$$A + B = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

and the product AB *is not defined.*

One can also define a scalar multiplication which, together with the matrix addition defined above, gives $\mathcal{M}_{m\times n}(R)$ a vector space structure. Let $\alpha \in R$ and let $A = [a_{i,j}]$ be a $m \times n$ matrix over R. Then, the scalar multiplication of α and A, the matrix $C = [c_{i,j}]$, is given by

$$c_{i,j} = \alpha a_{i,j}.$$

If A is an $m \times n$ matrix, then the *transpose* matrix of A is denoted by A^T and it is the $n \times m$ matrix whose (i, j)th entry is the same as the (j, i)th entry of the original matrix A.

Example 2.4.5. *Let A and B be the following matrices over $\mathbb{R}$:*

$$A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \textit{and} \quad B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.$$

Then,

$$A^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad \textit{and} \quad B^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

For a matrix A, the submatrix $A_{i,j}$ is obtained by deleting the ith row and the jth column.

Example 2.4.6. *Consider the matrix B of the previous example. Then $B_{1,2} = \begin{bmatrix} 4 & 6 \end{bmatrix}$.*

With each $n \times n$ matrix $A = [a_{i,j}]$ there is associated a unique number called the *determinant* of A and written $\det(A)$ or $|A|$. The determinant of A can be computed recursively as follows:

1. $|A| = a_{1,1}$, if $n = 1$;
2. $|A| = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$, if $n = 2$;
3. $|A| = \sum_{j=1}^{n} (-1)^{1+j} a_{1,j}|A_{1,j}|$, if $n > 2$.

It is well known that a matrix $A \in \mathcal{M}_n$ has rank n, i.e., maximal rank, if and only if its determinant is not zero [McC71].

For a $n \times n$ matrix A, the *adjoint matrix* of A is the matrix

$$adj(A) = [c_{i,j}],$$

where

$$c_{i,j} = (-1)^{i+j} \det(A_{j,i}).$$

Example 2.4.7. *Consider the matrices*

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \textit{and} \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$

defined over $\mathbb{F}_2$. *Then,* $\det(A) = 1$, $\det(B) = 0$,

$$adj(A) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \textit{and} \quad adj(B) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let A be a $n \times n$ matrix. A is called *invertible* (also *non-singular*) if there exists a $n \times n$ matrix B such that

$$AB = BA = I_n.$$

If this is the case, the matrix B is uniquely determined by A and is called the *inverse* of A, denoted by A^{-1}. The inverse of A can be computed in several ways. For example,

$$A^{-1} = \frac{1}{\det(A)} adj(A).$$

Furthermore, A is invertible if and only if $\det(A) \neq 0$ or, equivalently, $rank(A) = n$ [McC71]. The set of all $n \times n$ invertible matrices over R is denoted by $GL_n(R)$, which stands for general linear group of degree n over R.

Example 2.4.8. *The matrix* B *of the previous example is not invertible, while the matrix* A *is invertible and* $A^{-1} = adj(A)$.

Notice that non-square matrices are not invertible. However, they can be left or right invertible. A $m \times n$ matrix A is *left* (*right*) *invertible* if there is a $n \times m$ matrix B such that $BA = I_n$ ($AB = I_m$). Such matrix B is called a *left* (*right*) *inverse* of A. One knows that A is left (right) invertible if and only if $rank(A) = n$ ($rank(A) = m$), i.e., the columns (rows) of A are linearly independent.

One says that a matrix is in *reduced row echelon form* if and only if all the following conditions hold:

- the first non-zero entry in each row is 1;
- each row has its first non-zero entry in a later column than any previous rows;
- all entries above and below the first non-zero entry of each row are zero;
- all rows having nothing but zeros are below all other rows of the matrix.

The matrix is said to be in *reduced column echelon form* if its transpose matrix is in reduced row echelon form.

Example 2.4.9. *The following matrix over $\mathbb{F}_2$ is in reduced row echelon form but is not in reduced column echelon form:*

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.5 Linear Maps

Let V and W be vector spaces over the same field $\mathbb{F}$. A mapping $f : V \to W$ is called a *linear transformation*, *linear map* or an *homomorphism* of V into W, if the following conditions are true:

- $f(v_1 + v_2) = f(v_1) + f(v_2)$, for all v_1, v_2 in V;
- $f(\alpha v) = \alpha f(v)$, for all α in $\mathbb{F}$ and for all v in V.

The first condition states that addition is preserved under the mapping f. The second asserts that also scalar multiplication is preserved under the mapping f. This is equivalent to require that the same happens for any linear combination of vectors, i.e., that for any vectors $v_1, \ldots, v_n \in V$, and scalars $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$, the following equality holds:

$$f(\alpha_1 v_1 + \cdots + \alpha_n v_n) = \alpha_1 f(v_1) + \cdots + \alpha_n f(v_n).$$

Denoting the zero elements of the vector spaces V and W by 0_V and 0_W respectively, it follows that $f(0_V) = 0_W$ because, letting $\alpha = 0$ in the second condition, one gets:

$$f(0_V) = f(0 \cdot 0_V) = 0 f(0_V) = 0_W.$$

An homomorphism which is a bijective mapping is called a *linear isomorphism*, and if there exists an isomorphism φ of V onto W it is said that V is *isomorphic* to W, denoted by $V \simeq W$, and φ is called a *vector space isomorphism*.

Example 2.5.1. *Let $f : \mathbb{F}_2^3 \to \mathbb{F}_2^2$ be the mapping defined by:*

$$f(x, y, z) = (x + y, z).$$

Let us show that f is a linear map.

1. *Let $v = (v_1, v_2, v_3), w = (w_1, w_2, w_3) \in \mathbb{F}_2^3$. Then*

$$\begin{aligned} f(v + w) &= f(v_1 + w_1, v_2 + w_2, v_3 + w_3) \\ &= (v_1 + w_1 + v_2 + w_2, v_3 + w_3) \\ &= (v_1 + v_2, v_3) + (w_1 + w_2, w_3) \\ &= f(v) + f(w). \end{aligned}$$

2. *Let $\alpha \in \mathbb{F}_2$ and $v = (v_1, v_2, v_3) \in \mathbb{F}_2^3$. Then*

$$\begin{aligned} f(\alpha v) &= f(\alpha v_1, \alpha v_2, \alpha v_3) \\ &= (\alpha v_1 + \alpha v_2, \alpha v_3) \\ &= \alpha(v_1 + v_2, v_3) \\ &= \alpha f(v). \end{aligned}$$

Since addition and scalar multiplication are preserved under f, one concludes that f is a linear map.

2.6 Graphs

A *directed graph* is an ordered pair (V, Γ) where V is called the *vertex set* and $\Gamma \subseteq V \times V$ is called the *arc set*. If $V = \emptyset$ the graph is called the *empty graph*. Elements in V are called *vertices* and elements in Γ are called *arcs*. For an arc $u = (a, b) \in \Gamma$, a is called the *initial vertex* of u and b the *terminal vertex*.

A *path* of the graph (V, Γ) is a finite or infinite sequence of arcs where the terminal vertex of an arc is the initial vertex of the next arc. The number of arcs in the sequence is called the *length* of the path.

If $\varphi = u_1 u_2 \cdots u_n$ is a path of the graph and the terminal vertex of the arc u_n is the inicial vertex of u_1, the path φ is called a *circuit*. Evidently, if there exists a circuits then there exists a path of infinite length.

The *levels of vertices* can be defined recurrently as follows:

- For any vertex $a \in V$, if a is not a terminal vertex of any arc then the level of a is defined to be 0;
- For any vertex a in V, if the levels of all initial vertices of arcs with a as terminal vertex have been defined and the maximum is h then the level of a is defined to be $h + 1$.

If the level of each vertex of (V, Γ) is defined and the maximum is ℓ, the graph *has level* and the *level of the graph* is defined to be ℓ. Clearly, if each vertex of the graph is an isolated vertex (i.e, vertex with level 0) then the level of the graph is 0. The level of the empty graph is defined to be -1.

Chapter 3

Finite Transducers

3.1 Preliminaries on Finite Transducers

An alphabet is a non-empty finite set of elements where the elements are called *symbols* or *letters*. Given an alphabet A, a finite sequence of symbols from A, say $\alpha = a_0a_1\cdots a_{l-1}$, is called a *word* over A, and l its *length* which is denoted by $|\alpha|$. The *empty word* is a word of length $l = 0$, i.e, the empty sequence, denoted by ε. Let A^n be the set of words of length n, where $n \in \mathbb{N}_0$, then, for example, $A^0 = \{\varepsilon\}$. Let $A^\star = \cup_{n\geq 0}A^n$ be the set of all finite words and $A^\omega = \{a_0a_1\cdots a_n\cdots \mid a_i \in A\}$ be the set of infinite words. The concatenation of two words in A^*, say $\alpha = a_0a_1\cdots a_{m-1}$ and $\beta = b_0b_1\cdots b_{n-1}$, is also a word in A^* of length $m+n$ and is denoted by $\alpha\beta$. Similarly, if $\alpha = a_0a_1\cdots a_{m-1} \in A^*$ and $\beta = b_0b_1\cdots b_{n-1}\cdots \in A^\omega$, then the concatenation of α and β is the element $a_0a_1\cdots a_{m-1}b_0b_1\cdots b_{n-1}\cdots$ of A^ω.

In the context of this work, a *finite transducer* (FT) is a finite state sequential machine which, in any given state, reads a symbol from a set $\mathcal{X}$, produces a symbol from a set $\mathcal{Y}$, and switches to another state. Thus, given an initial state and a finite input sequence, a transducer produces an output sequence of the same length. The formal definition of a finite transducer is the following.

Definition 3.1.1. *A* finite transducer *is a quintuple* $\langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda\rangle$*, where:*

- $\mathcal{X}$ *is a nonempty finite set called the* input alphabet*;*
- $\mathcal{Y}$ *is a nonempty finite set called the* output alphabet*;*

- S *is a nonempty finite set called the* set of states*;*
- $\delta : S \times \mathcal{X} \to S$ *called the* state transition function*; and*
- $\lambda : S \times \mathcal{X} \to \mathcal{Y}$ *called the* output function*.*

Any state of S can be used as the initial state. In these transducers, for each state and input, only one output is possible, therefore, they are deterministic.

Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. The state transition function δ and the output function λ can be extended to finite words, i.e., elements of $\mathcal{X}^\star$, recursively, as follows:

$$\begin{aligned} \delta(s, \varepsilon) &= s & \qquad \delta(s, x\alpha) &= \delta(\delta(s, x), \alpha) \\ \lambda(s, \varepsilon) &= \varepsilon & \qquad \lambda(s, x\alpha) &= \lambda(s, x)\, \lambda(\delta(s, x), \alpha), \end{aligned}$$

where $s \in S$, $x \in \mathcal{X}$, and $\alpha \in \mathcal{X}^\star$.

Example 3.1.2. *Let* $M = \langle \{0, 1\}, \{a, b\}, \{s_1, s_2\}, \delta, \lambda \rangle$ *be the transducer defined by:*

$$\begin{aligned} &\delta(s_1, 0) = s_1, && \delta(s_1, 1) = s_2, && \delta(s_2, 0) = s_1, && \delta(s_2, 1) = s_2, \\ &\lambda(s_1, 0) = a, && \lambda(s_1, 1) = a, && \lambda(s_2, 0) = b, && \lambda(s_2, 1) = b. \end{aligned}$$

Then, for example,

$$\delta(s_1, 01) = \delta(\delta(s_1, 0), 1) = \delta(s_1, 1) = s_2,$$

$$\lambda(s_1, 01) = \lambda(s_1, 0)\lambda(\delta(s_1, 0), 1) = a\lambda(s_1, 1) = aa,$$

and

$$\delta(s_1, 0010110) = s_1,$$

$$\lambda(s_1, 0010110) = aaababb.$$

Example 3.1.3. *Let* $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^3, \mathbb{F}_2^2, \delta, \lambda \right\rangle$ *be the transducer defined by:*

$$\delta(s, x) = As + Bx,$$

$$\lambda(s, x) = Cs + Dx,$$

for all $s \in \mathbb{F}_2^2$, $x \in \mathbb{F}_2^2$ *and where*

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},\ C = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \text{ and } D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Take $s = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ *and* $\alpha = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. *Then,*

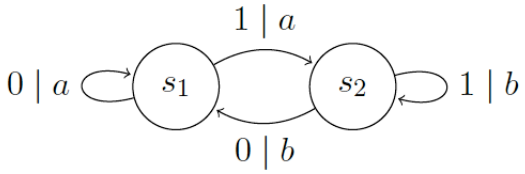
$$\delta(s, \alpha) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\lambda(s, \alpha) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

M *is what is called a linear finite transducer. The formal definition will be given later.*

A transducer can be represented by a diagram that is a digraph with labeled nodes and arcs, where loops and multiple arcs are allowed. Each state of the tranducer is represented by a node and each arc indicates a transition between states. The label of each arc is a compound symbol of the form $i|o$, where i and o stand for the input and output symbol, respectively. This representation is useful to deal by hand with the computations of some examples presented in this chapter.

Example 3.1.4. *The transducer* M *defined in* Example *3.1.2 is represented by the diagram below:*



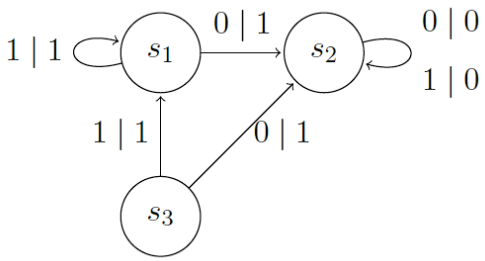
Given this diagram, it is quite easy to compute $\delta(s, \alpha)$ *and* $\lambda(s, \alpha)$ *for the transducer, where* $s \in S$ *and* $\alpha \in \mathcal{X}^*$.

Definition 3.1.5. *Let* $M_1 = \langle \mathcal{X}, \mathcal{Y}_1, S_1, \delta_1, \lambda_1 \rangle$ *and* $M_2 = \langle \mathcal{X}, \mathcal{Y}_2, S_2, \delta_2, \lambda_2 \rangle$ *be two finite transducers. Let* $s_1 \in S_1$ *and* $s_2 \in S_2$. *One says that* s_1 *and* s_2 *are* equivalent, *and denote this relation by* $s_1 \sim s_2$, *if*

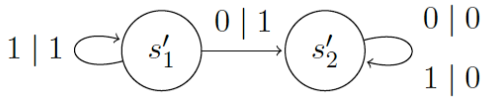
$$\forall \alpha \in \mathcal{X}^*, \lambda_1(s_1, \alpha) = \lambda_2(s_2, \alpha).$$

It is obvious that if $s_1 \sim s_2$ then $\forall x \in \mathcal{X}, \delta_1(s_1, x) \sim \delta_2(s_2, x)$.

Example 3.1.6. *Let $M = \langle \mathbb{F}_2, \mathbb{F}_2, \{s_1, s_2, s_3\}, \delta, \lambda \rangle$ be the transducer induced by the diagram:*



and let $M' = \langle \mathbb{F}_2, \mathbb{F}_2, \{s'_1, s'_2\}, \delta', \lambda' \rangle$ be the transducer induced by:



Then

- $s_2 \sim s'_2$, *because* $\forall \alpha \in \mathcal{X}^*, \lambda(s_2, \alpha) = 0 \cdots 0 = \lambda'(s'_2, \alpha)$;
- $s_1 \sim s_3 \sim s'_1$

To prove that $s_1 \sim s_3$, let α be a non-empty word in $\mathbb{F}_2^*$. Then, either α is of the form 0β or α is of the form 1β, for some β in $\mathbb{F}_2^*$. In the first case, one has

$$\lambda(s_1, 0\beta) = \lambda(s_1, 0)\lambda(\delta(s_1, 0), \beta) = 1\lambda(s_2, \beta),$$

and

$$\lambda(s_3, 0\beta) = \lambda(s_3, 0)\lambda(\delta(s_3, 0), \beta) = 1\lambda(s_2, \beta),$$

It follows that $\lambda(s_1, 0\beta) = \lambda(s_3, 0\beta)$, for all $\beta \in \mathcal{X}^*$. Analogously,

$$\lambda(s_1, 1\beta) = 1\lambda(s_1, \beta) = \lambda(s_3, 1\beta),$$

for all $\beta \in \mathcal{X}^*$. Therefore, $\forall \alpha \in \mathcal{X}^*, \lambda(s_1, \alpha) = \lambda(s_3, \alpha)$, i.e., $s_1 \sim s_3$. It is also easy to see that $s_1 \sim s'_1$.

The definition of *equivalent states* can be used to define *equivalent transducers*.

Definition 3.1.7. *Let* $M_1 = \langle \mathcal{X}, \mathcal{Y}_1, S_1, \delta_1, \lambda_1 \rangle$ *and* $M_2 = \langle \mathcal{X}, \mathcal{Y}_2, S_2, \delta_2, \lambda_2 \rangle$ *be two finite transducers.* M_1 *and* M_2 *are said to be* equivalent, *and denote this by* $M_1 \sim M_2$, *if the following two conditions are simultaneously satisfied:*

- $\forall s_1 \in S_1, \exists s_2 \in S_2 : s_1 \sim s_2$;
- $\forall s_2 \in S_2, \exists s_1 \in S_1 : s_1 \sim s_2$.

The relation $\sim$ *defines an equivalence relation on the set of finite transducers.*

Example 3.1.8. *The transducers* M *and* M' *of* Example *3.1.6 are equivalente since* $s_1 \sim s_3 \sim s'_1$ *and* $s_2 \sim s'_2$.

3.2 Concepts on Invertibility

A fundamental concept in this work is the concept of injectivity that is behind the invertibility property of the transducers used for cryptographic purposes. In fact, it will be presented two concepts: the concept of ω-injectivity and the concept of injectivity with a certain delay. These two notions of injectivity were introduced by Tao, who called them *weakly invertible* and *weakly invertible with a certain delay*, respectively [Tao09]. Here it will be used names that are more naturally related to how these terms are used in other mathematical settings.

Definition 3.2.1. *A finite transducer* $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ *is* ω-injective, *if*

$$\forall s \in S,\ \forall \alpha, \alpha' \in \mathbb{X}^\omega, \quad \lambda(s, \alpha) = \lambda(s, \alpha') \implies \alpha = \alpha'.$$

That is, for any $s \in S$, *and any* $\alpha \in \mathcal{X}^\omega$, α *can be uniquely determined by* s *and* $\lambda(s, \alpha)$.

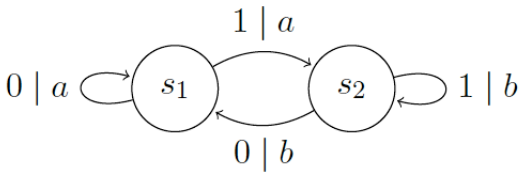
Definition 3.2.2. *A finite transducer* $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ *is* injective with delay τ, *or* τ-injective, *with* $\tau \in \mathbb{N}_0$, *if*

$$\forall s \in S,\ \forall x, x' \in \mathcal{X},\ \forall \alpha, \alpha' \in \mathcal{X}^\tau, \quad \lambda(s, x\alpha) = \lambda(s, x'\alpha') \implies x = x'.$$

That is, for any $s \in S$, $x \in \mathcal{X}$, *and any* $\alpha \in \mathcal{X}^\tau$, x *can be uniquely determined by* s *and* $\lambda(s, x\alpha)$.

If a transducer is injective with delay 0, given the initial state and the output symbol, one can recover the input symbol used. If a transducer is injective with some delay τ, $\tau \in \mathbb{N}$, the first symbol of an input sequence of length $\tau + 1$ can be recovered, given the initial state and the output sequence. Obviously, if the input sequence has length $\tau + \ell$, $\ell \in \mathbb{N}$, one can recover the first ℓ input symbols.

Example 3.2.3. *The transducer presented in* Example *3.1.2 and which is represented by the diagram*



is injective with delay 1*. To prove that, one has to compute the output for every state and every input sequence of length* 2*:*

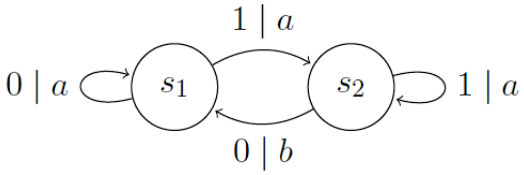
$$\lambda(s_1, 00) = aa, \quad \lambda(s_2, 00) = ba, \quad \lambda(s_1, 10) = ab, \quad \lambda(s_2, 10) = bb,$$
$$\lambda(s_1, 01) = aa, \quad \lambda(s_2, 01) = ba, \quad \lambda(s_1, 11) = ab, \quad \lambda(s_2, 11) = bb.$$

From these outputs, one can conclude that

$$\forall s \in \{s_1, s_2\},\ \forall x_0x_1, x_0'x_1' \in \{0,1\}^2, \quad \lambda(s, x_0x_1) = \lambda(s, x_0'x_1') \implies x_0 = x_0',$$

which proves, by definition, that the transducer is injective with delay 1*. Moreover, the transducer is not injective with delay* 0 *(for example,* $\lambda(s_1, 0) = a = \lambda(s_1, 1)$ *and* $0 \neq 1$*).*

Example 3.2.4. *The transducer* $M = \langle \{0,1\}, \{a,b\}, \{s_1, s_2\}, \delta, \lambda \rangle$ *induced by the diagram*



is not injective with delay 1 *since, for example,* $\lambda(s_1, 01) = \lambda(s_1, 11)$ *and* $0 \neq 1$*.*

Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. Clearly, if M is injective with delay $\tau \in \mathbb{N}_0$ then M is also injective with delay k, for $k \geq \tau$, which implies that it is also ω-injective. Tao proved that the converse is also true [Tao09, Corollary 1.4.3]. To demonstrate this result consider the graph $G_M = (V, \Gamma)$ constructed from M as follows. Let

$$R = \{(\delta(s,x), \delta(s,x')) \mid x \neq x',\ \lambda(s,x) = \lambda(s,x'),\ x, x' \in \mathcal{X},\ s \in S\}.$$

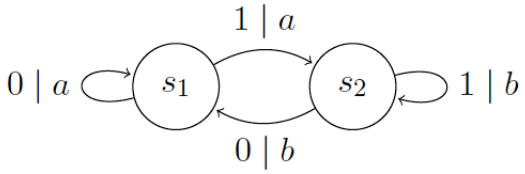
Obviously, if $(s, s') \in R$ then $(s', s) \in R$. If $R = \emptyset$ then G_M is the empty graph. In the case of $R \neq \emptyset$, let the vertex set V of G_M be the minimal subset of $S \times S$ satisfying the following conditions:

- $R \subseteq V$;
- $(s, s') \in V \wedge \lambda(s,x) = \lambda(s',x') \implies (\delta(s,x), \delta(s',x')) \in V$, where $x, x' \in \mathcal{X}$.

Let the arc set Γ of G_M be the set of all arcs $((s,s'), (\delta(s,x), \delta(s',x')))$ satisfying:

- $(s, s') \in V$;
- $\lambda(s,x) = \lambda(s',x')$, where $x, x' \in \mathcal{X}$.

Example 3.2.5. *Consider the transducer represented by the diagram*

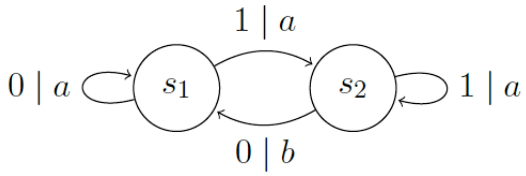


To construct the graph G_M, first one has to construct the set R defined as above. In this transducer one has:

$$\lambda(s_1, 0) = \lambda(s_1, 1) = a$$
$$\lambda(s_2, 0) = \lambda(s_2, 1) = b$$

then $R = \{(\delta(s_1,0), \delta(s_1,1)); (\delta(s_1,1), \delta(s_1,0)); (\delta(s_2,0), \delta(s_2,1)); (\delta(s_2,1), \delta(s_2,0))\} = \{(s_1,s_2), (s_2,s_1)\}$. To construct the vertex set V one has to initialize it as $V = R$ then, for all pairs $(s,s') \in V$ that produce the same output for some $x, x' \in \{0,1\}$, $(\delta(s,x), \delta(s',x')) \in V$. Since $\forall x \in \{0,1\}$, $\lambda(s_1,x) = a$ and $\lambda(s_2,x) = b$, the pairs in R never produce the same output, then $V = R = \{(s_1,s_2), (s_2,s_1)\}$. Therefore, the graph G_M is composed by two isolated vertices.

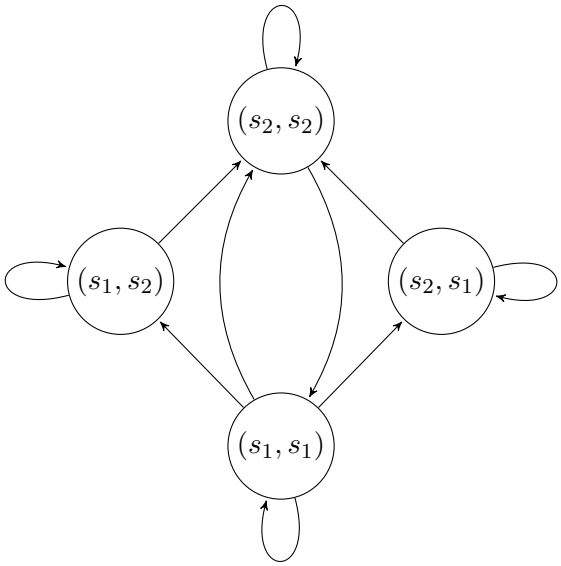
Example 3.2.6. *Consider the transducer M presented in 3.2.4 induced by the diagram*



The transducer produces the output a when the state is s_1 or when the state is s_2 and the input is 1. Therefore, since R is composed by the pairs of the states transition obtained with differents inputs that produce the same output, $R = \{(\delta(s_1, 0), \delta(s_1, 1)); (\delta(s_1, 1), \delta(s_1, 0)); (\delta(s_1, 0), \delta(s_2, 1)); (\delta(s_2, 1), \delta(s_1, 0))\} = \{(s_1, s_2), (s_2, s_1)\}$.

Let the vertex set V be equal to R. Since $\lambda(s_1, 0) = \lambda(s_2, 1)$ and $\lambda(s_1, 1) = \lambda(s_2, 1)$, the vertex (s_1, s_2) has an arc to the vertex $(\delta(s_1, 0), \delta(s_2, 1)) = (s_1, s_2)$ and an arc to the vertex $(\delta(s_1, 1), \delta(s_2, 1)) = (s_2, s_2)$. Analogously, (s_2, s_1) has an arc to itself and an arc to the vertex (s_2, s_2). The new vertex (s_2, s_2), obviously, has an arc to itself and an arc to (s_1, s_1), since $\delta(s_2, 0) = s_1$. Finally, (s_1, s_1) has an arc to all the vertices since $\lambda(s_1, 0) = \lambda(s_1, 1)$ and $\delta(s_1, 0) = s_1 \wedge \delta(s_1, 0) = s_2$.

The graph G_M is represented by:



The following theorems prove that if M is ω-injective then M is injective with some delay τ.

Theorem 3.2.7. *Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. M is ω-injective if and only if the graph G_M has no circuit. Moreover, if G_M has no circuit and its level is ρ then the minimum τ such that M is τ-injective is $\rho + 1$.*

Proof. Suppose that G_M has a circuit. From the construction of the graph, there exists a path $u_1u_2\cdots u_k$ such that the initial vertex of u_1 is in R and $u_ru_{r+1}\cdots u_k$ is a circuit for some $r, 1 \leq r \leq k$ (which means that the terminal vertex of u_k is the initial vertex of u_r). One has that $\forall i \in \{1, \cdots, k\}$, $u_i = ((s_i, s_i'), (\delta(s_i, x_i), \delta(s_i', x_i')))$ and $\lambda(s_i, x_i) = \lambda(s_i', x_i')$ for some $x_i, x_i' \in \mathcal{X}$. Since the inicial vertex of u_1 is in R, there exists $x_0, x_0' \in \mathcal{X}$, $s_0 \in S$ such that $\lambda(s_0, x_0) = \lambda(s_0, x_0')$ and $x_0 \neq x_0'$. Taking $\alpha = x_0x_1\cdots x_{r-1}x_r\cdots x_kx_r\cdots x_k\cdots$ and $\alpha' = x_0'x_1'\cdots x_{r-1}'x_r'\cdots x_k'x_r'\cdots x_k'\cdots$, one has that $\alpha \neq \alpha'$ (since $x_0 \neq x_0'$) and $\lambda(s_0, \alpha) = \lambda(s_0, \alpha')$, thus M is not ω-injective (by *Definition* 3.2.1).

Conversely, suppose that G_M has no circuit. Then G_M has level. Let ρ be the level of the graph, where $\rho \in \mathbb{N}_0 \cup \{-1\}$. In the case of $R = \emptyset$, it is obvious that $\rho = -1$ and M is injective with delay 0 $(= \rho + 1)$ since there not exists $s \in S$ such that $\lambda(s, x) = \lambda(s, x')$ and $x \neq x'$. In the case of $R \neq \emptyset$, for any state s_0 of M and for any input sequence $\alpha = x_0x_1\cdots x_{\rho+1}$ and $\alpha' = x_0'x_1'\cdots x_{\rho+1}'$, reduction to absurdity proves that $\lambda(s_0, \alpha) = \lambda(s_0, \alpha') \Longrightarrow x_0 = x_0'$:

> Suppose that $\lambda(s_0, x_0x_1\cdots x_{\rho+1}) = \lambda(s_0, x_0'x_1'\cdots x_{\rho+1}')$ and $x_0 \neq x_0'$ for some $s_0 \in S$ and some input letters. Since $\lambda(s_0, x_0x_1\cdots x_{\rho+1}) = \lambda(s_0, x_0'x_1'\cdots x_{\rho+1}')$, one has that $\lambda(s_0, x_0) = \lambda(s_0, x_0')$ and $\lambda(s_i, x_i) = \lambda(s_i', x_i')$, $i = 1, 2, \cdots, \rho + 1$. Since $x_0 \neq x_0'$, $(s_1, s_1') = (\delta(s_0, x_0), \delta(s_0, x_0')) \in R$. Moreover, for any i, $1 \leq i \leq \rho + 1$, there exists an arc $u_i = ((s_i, s_i'), (s_{i+1}, s_{i+1}'))$. Thus $u_1u_2\cdots u_{\rho+1}$ is a path of G_M. This means that the level of the graph is at least $\rho + 1$ which contradicts that the level of G_M is ρ.

Since $\lambda(s_0, \alpha) = \lambda(s_0', \alpha') \Longrightarrow x_0 = x_0'$, this proves that M is injective with delay $\rho + 1$ (by *Definition* 3.2.2). □

Example 3.2.8. *The transducer presented in* Example *3.2.5 has a graph G_M with two isolated vertices, thus G_M has level 0, which means that the transducer is injective with delay 1 (as seen in* Example *3.2.3). The graph of the transducer in* Example *3.2.6 has a circuit, therefore this transducer is not ω-injective.*

Corollary 3.2.9. *Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. If M is ω-injective, then there exists a non-negative integer $\tau \leq \frac{|S|(|S|-1)}{2}$ such that M is injective with delay τ.*

Proof. Suppose that M is ω-injective. If G_M is the empty graph then $R = \emptyset$, which means that $\forall x, x' \in \mathcal{X},\ \forall s \in S, \quad x \neq x' \Longrightarrow \lambda(s, x) \neq \lambda(s, x')$. Since the statement is logically equivalent to $\forall x, x' \in \mathcal{X},\ \forall s \in S \quad \lambda(s, x) = \lambda(s, x') \Longrightarrow x = x'$ it follows, from *Definition* 3.2.1, that M is injective with delay 0 and $0 \leq \frac{|S|(|S|-1)}{2}$.

Conversely, suppose that G_M is not the empty graph. Then, from the previous theorem, G_M has no circuits (M is ω-injective). If $\exists s \in S$ such that $(s, s) \in V$, then $(s', s') = (\delta(s, x), \delta(s, x)) \in V$, since $\forall x \in \mathcal{X},\ \lambda(s, x) = \lambda(s, x)$. This yields that $s_1 \neq s_2$ for any $(s_1, s_2) \in V$. Thus, $|V| \leq |S|(|S| - 1)$. It is evident that $(s_1, s_2) \in V$ if and only if $(s_2, s_1) \in V$, and that $((s_1, s_2), (s_3, s_4)) \in \Gamma$ if and only if $((s_2, s_1), (s_4, s_3)) \in \Gamma$. Therefore, the number of vertices with level i, $0 \leq i \leq \rho$, is at least 2. Since the number of levels is $\rho + 1$, one has that $2(\rho + 1) \leq |S|(|S| - 1) \Leftrightarrow \rho + 1 \leq \frac{|S|(|S|-1)}{2}$. From *Theorem* 3.2.7, $\tau = \rho + 1$, thus $\tau \leq \frac{|S|(|S|-1)}{2}$. □

Example 3.2.10. *From the previous theorem one may conclude, again, that the transducer M defined in* Example *3.2.4 is not ω-injective, since it is not injective with delay 1 and the set of states has size 2* $\left(\tau \leq \frac{|S|(|S|-1)}{2} = \frac{2\times 1}{2} = 1\right)$.

Naturally, injective transducers should have inverses of some sort. In order to describe the appropriate concept, the following definition introduces a notion of an inverse state of a given state.

Definition 3.2.11. *Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ and $M' = \langle \mathcal{Y}, \mathcal{X}, S', \delta', \lambda' \rangle$ be two finite transducers. Let $s \in S$ and $s' \in S'$, then s' inverts s with delay $\tau \in \mathbb{N}_0$ or s' is an inverse state with delay τ of s when*

$$\forall \alpha \in \mathcal{X}^{\omega},\ \lambda'(s', \lambda(s, \alpha)) = \gamma\alpha, \text{ for some } \gamma \in \mathcal{X}^{\tau}.$$

Remark 3.2.12. *In the previous definition one may replace $\mathcal{X}^{\omega}$ by $\mathcal{X}^{*}$, but then one has to replace $\lambda'(s', \lambda(s, \alpha)) = \gamma\alpha$ by $\lambda'(s', \lambda(s, \alpha)) = \gamma\alpha'$, where α' consists of the first $|\alpha| - \tau$ characters of α.*

Basically, using an inverse state s' with delay τ of a given state s one can start to recover the input symbols of M after τ symbols being read by M'.

The figure below gives a schematic representation of the concept of inverse state with delay τ, where $\alpha = x_1x_2\ldots$ and $\lambda(s,\alpha) = y_1y_2\ldots$:

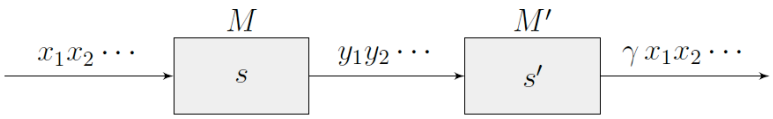
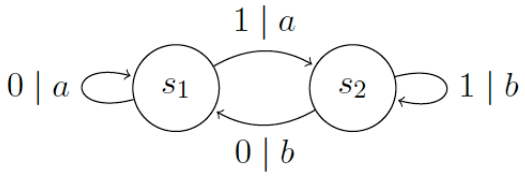
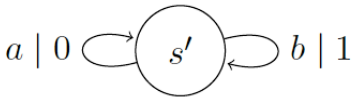


Figure 3.1: Concept of Inverse State with delay τ

Example 3.2.13. *Let* $M = \langle\{0,1\},\{a,b\},\{s_1,s_2\},\delta,\lambda\rangle$ *be the transducer induced by the diagram:*



and let $M' = \langle\{a,b\},\{0,1\},\{s'\},\delta',\lambda'\rangle$ *be the transducer:*



The state s' *of* M' *inverts the states* s_1 *and* s_2 *of* M *with delay* 1. *To prove that, it is enough to show that, for all* $x_1x_2 \in \{0,1\}^2$ *and for all* $s \in \{s_1,s_2\}$, *one has*

$$\lambda'(s',\lambda(s,x_1x_2)) = xx_1, \text{ for some } x \in \{0,1\},$$

because this implies that for all $\alpha \in \{0,1\}^\omega$ *and for all* $s \in \{s_1,s_2\}$,

$$\lambda'(s',\lambda(s,\alpha)) = x\alpha, \text{ for some } x \in \{0,1\}.$$

Using the diagrams of the tranducers one easily gets

$$\lambda'(s',\lambda(s_1,00)) = \lambda'(s',aa) = 00, \qquad \lambda'(s',\lambda(s_1,10)) = \lambda'(s',ab) = 01,$$
$$\lambda'(s',\lambda(s_1,01)) = \lambda'(s',aa) = 00, \qquad \lambda'(s',\lambda(s_1,11)) = \lambda'(s',ab) = 01,$$
$$\lambda'(s',\lambda(s_2,00)) = \lambda'(s',ba) = 10, \qquad \lambda'(s',\lambda(s_2,10)) = \lambda'(s',bb) = 11,$$
$$\lambda'(s',\lambda(s_2,01)) = \lambda'(s',ba) = 10, \qquad \lambda'(s',\lambda(s_2,11)) = \lambda'(s',bb) = 11.$$

This proves that s' *inverts the states* s_1 *and* s_2 *with delay* 1.

Definition 3.2.14. *Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. One says that M is left invertible with delay τ if there is a transducer $M' = \langle \mathcal{Y}, \mathcal{X}, S', \delta', \lambda' \rangle$ such that*

$$\forall s \in S,\ \exists s' \in S',\ s' \text{ inverts } s \text{ with delay } \tau.$$

The transducer M' is called a left inverse with delay τ of M.

It is clear that, in the previous example, the transducer M' is a left inverse with delay 1 of M.

If M' is a left inverse with delay τ of M, then M' can recover the input of M with a delay of τ input symbols.

The following result, proven by Tao, establishes the fundamental relation between the injectivity of a transducer and the existence of a left inverse. [Tao09]

Theorem 3.2.15. *A finite transducer $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ is injective with delay τ if and only if there exists a finite tranducer $M' = \langle \mathcal{Y}, \mathcal{X}, S', \delta', \lambda' \rangle$ such that M' is a left inverse with delay τ of M.*

Later, in this work [*Chapter* 4], it will be presented a necessary and sufficient condition to the transducers used in the FAPKC to be invertible (transducers with memory). Furthermore, it will be shown a method to construct a left inverse of a transducer.

3.3 The notion of Linear Finite Transducer

Definition 3.3.1. *If $\mathcal{X}$, $\mathcal{Y}$ and S are vector spaces over a field $\mathbb{F}$ and both $\delta : S \times \mathcal{X} \to S$ and $\lambda : S \times \mathcal{X} \to \mathcal{Y}$ are bilinear maps, then $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ is called a* linear finite transducer (LFT) *over $\mathbb{F}$ and the size of M, denoted $size(M)$, is the dimension of S as a vector space.*

Example 3.3.2. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^2, \mathbb{F}_2^2, \delta, \lambda \rangle$ be the transducer defined by:*

$$\delta(s, x) = (s_2 + x_1, s_1 + x_2 + x_3),$$
$$\lambda(s, x) = (s_1 + x_1 + x_3, s_2 + x_2),$$

for all $s = (s_1, s_2) \in \mathbb{F}_2^2$ and for all $x = (x_1, x_2, x_3) \in \mathbb{F}_2^3$. The state transition function $\delta : \mathbb{F}_2^5 \to \mathbb{F}_2^2$ and the output function $\lambda : \mathbb{F}_2^5 \to \mathbb{F}_2^2$ are linear maps, therefore, M is a linear finite

transducer over $\mathbb{F}_2$ and the size of M is $dim(\mathbb{F}_2^2) = 2$. Moreover, if one considers the standard bases of $\mathbb{F}_2^5$ and $\mathbb{F}_2^2$, those maps are represented in terms of matrices in the following way

$$\begin{aligned}\delta(s,x) &= \begin{bmatrix}0 & 1 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 1\end{bmatrix}\begin{bmatrix}s_1 & s_2 & x_1 & x_2 & x_3\end{bmatrix}^T\\ &= \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}s_1\\ s_2\end{bmatrix} + \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}\\ &= \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}s + \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\end{bmatrix}x,\end{aligned}$$

$$\begin{aligned}\lambda(s,x) &= \begin{bmatrix}1 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}s_1 & s_2 & x_1 & x_2 & x_3\end{bmatrix}^T\\ &= \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}s_1\\ s_2\end{bmatrix} + \begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}\\ &= \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}s + \begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}x,\end{aligned}$$

Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda\rangle$ be a linear finite transducer over a field $\mathbb{F}$. If $\mathcal{X}$, $\mathcal{Y}$ and S have dimensions ℓ, m and n, respectively, then there exist matrices $A \in \mathcal{M}_n(\mathbb{F})$, $B \in \mathcal{M}_{n\times\ell}(\mathbb{F})$, $C \in \mathcal{M}_{m\times n}(\mathbb{F})$ and $D \in \mathcal{M}_{m\times\ell}(\mathbb{F})$, such that, in the appropriate bases,

$$\begin{aligned}\delta(s,x) &= As + Bx,\\ \lambda(s,x) &= Cs + Dx,\end{aligned}$$

for all $s \in S$, $x \in \mathcal{X}$. From the computations made on the previous example it is easy to understand how the matrices can be constructed from the maps δ and λ. The matrices A, B, C and D are called the *structural matrices* of M, and ℓ, m and n are called its *structural parameters*.

A linear finite transducer such that C is the null matrix (with the adequate dimensions) is called *trivial* since the output of this transducer only depends on the input.

3.4 Finite Transducers with memory

The *finite transducers with memory*, given a new input symbol, use the values of past inputs and, possibly, past outputs to compute a new output symbol. This type of transducers is the base of the cryptosystems being analyzed. The private keys are compounded by two finite transducers with memory, one linear and one quasi-linear. Quasi-linear finite transducers, which will be properly defined later in this section, are a special type of non-linear finite transducers that have a linear part and a non-linear part in a way that the invertibility is only affected by the linear part [Tao09]. The composition of these two types of transducers produces a non-linear transducer, the public key. The security of the FAPKCs rely on the difficulty of factoring this type of transducers, as well as, the difficulty of inverting them.

To be able to introduce the linear and quasi-linear finite transducers with memory, first, let us start by properly defining finite transducers with memory.

Let $\mathcal{X}$ be a non-empty set and $j \in \mathbb{N}$. Define $\sigma_j : \mathcal{X}^j \times \mathcal{X} \to \mathcal{X}^j$ by:

$$\sigma_j((x_1, x_2, \ldots, x_j), x) = (x_2, x_3, \ldots, x_j, x).$$

Definition 3.4.1. *Let $\phi : \mathcal{X}^{h+1} \times \mathcal{Y}^k \longrightarrow \mathcal{Y}$, with $h, k \in \mathbb{N}_0$ not simultaneously null, and $\mathcal{X}, \mathcal{Y}$ two non-empty finite sets. Let $M_\phi = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta_\phi, \lambda_\phi \rangle$ be the finite transducer such that, for all $x \in \mathcal{X}$, $\alpha \in \mathcal{X}^h$, $\beta \in \mathcal{Y}^k$, the state transition and output functions are given by:*

$$\delta_\phi(< \alpha, \beta >, x) =< \sigma_h(\alpha, x), \sigma_k(\beta, y) >,$$
$$\lambda_\phi(< \alpha, \beta >, x) = y,$$

where $y = \phi(\alpha, x, \beta)$ and $< \ldots >$ is used to denote the states of this transducer. M_ϕ is called the finite transducer with memory (h, k) defined by ϕ. If $k = 0$, then M_ϕ is said to be a finite transducer with input memory h.

As the name suggests, a finite transducer with memory is completely defined by its memory (h, k) and by the function ϕ. Notice that δ_ϕ and λ_ϕ are explicitly given by ϕ.

Below, there is a schematic representation of the state transition function for this kind of transducers. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a finite transducer with memory of order (h, k), with $h, k \in \mathbb{N}_0$ not simultaneously null. Consider the state $< x_1, x_2, \ldots, x_h, y_1, y_2, \ldots, y_k > \in \mathcal{X}^h \times \mathcal{Y}^k$ and let $y \in \mathcal{Y}$ be the output produced by M with the input symbol $x \in \mathcal{X}$. Then, the next state of M is given by:

$$< x_1, x_2, \ldots, x_h, y_1, y_2, \ldots, y_k > \xrightarrow{x \mid y} < x_2, \ldots, x_h, x, y_2, \ldots, y_k, y >$$

Notice that, the current state of M is composed by the last h input symbols and the last k output symbols.

Example 3.4.2. *Let M_ϕ be the finite transducer with memory of order $(3, 2)$ defined by the map $\phi : \mathbb{F}_2^6 \longrightarrow \mathbb{F}_2$ with $\phi(a, b, c, d, e, f) = ab + c + df$. Then $M_\phi = \langle \mathbb{F}_2, \mathbb{F}_2, \mathbb{F}_2^5, \delta_\phi, \lambda_\phi \rangle$ is such that*

$$\lambda_\phi(< x_1, x_2, x_3, y_1, y_2 >, x) = \phi(x_3, x_2, x_1, x, y_2, y_1), \text{ and}$$
$$\delta_\phi(< x_1, x_2, x_3, y_1, y_2 >, x) =< x_2, x_3, x, y_2, \lambda(< x_3, x_2, x_1, y_2, y_1 >, x) > .$$

Take $s =< 1, 1, 1, 1, 1 > \in \mathbb{F}_2^5$. Then,

$$\lambda_\phi(s, 0) = \phi(1, 1, 1, 0, 1, 1) = 0, \text{ and}$$
$$\delta_\phi(s, 0) =< 1, 1, 0, 1, 0 > .$$

In a finite transducer with memory of order (h, k), the output depends on the current input, the last h inputs and the last k outputs. Naturally, one needs to define an initial state. Usually, these transducers are defined by the infinite set of equations

$$y_{t+k} = \phi(x_{t+h}, x_{t+(h-1)}, \ldots, x_{t+1}, x_t, y_{t+(k-1)}, \ldots, y_{t+1}, y_t), \text{ for } t \geq 0,$$

starting with some initial state $< x_0, \ldots, x_{h-1}, y_0, \ldots, y_{k-1} >$. Notice that, if $k = 0$, the output only depends on the input, i.e., $y_t = \phi(x_{t+h}, x_{t+(h-1)}, \ldots, x_{t+1}, x_t)$, for $t \geq 0$, and some initial state $< x_0, \ldots, x_{h-1} >$.

For example, the transducer in the previous example could be defined as follows. Let $M_\phi = \langle \mathbb{F}_2, \mathbb{F}_2, \mathbb{F}_2^5, \delta_\phi, \lambda_\phi \rangle$ be the finite transducer with memory of order $(3, 2)$ defined by

$$y_{t+2} = \phi(x_{t+3}, x_{t+2}, x_{t+1}, x_t, y_{t+2}, y_{t+1}, y_t) = x_{t+3}x_{t+2} + x_{t+1} + x_t y_{t+1}, \text{ for } t \geq 0,$$

where $s =< x_0, x_1, x_2, y_0, y_1 >$ is the inicial state of the transducer. With this kind of notation we are assuming that

$$y_2y_3y_4\ldots = \lambda_\phi(< x_0, x_1, x_2, y_0, y_1 >, x_3x_4, x_5\ldots)$$

where $x_t, y_t \in \mathbb{F}_2$, for $t \geq 0$.

Example 3.4.3. *Let* $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^3, \mathbb{F}_2^2 \times (\mathbb{F}_2^3)^2, \delta, \lambda \right\rangle$ *be the finite transducer with memory of order* $(1,2)$ *defined by*

$$y_{t+2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} y_t, \text{ for } t \geq 0,$$

where $x_t \in \mathbb{F}_2^2$, $y_t \in \mathbb{F}_2^3$, *for* $t \geq 0$, *and* $< x_0, y_0, y_1 >$ *is the initial state of the transducer.*

Take $x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $y_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ *and* $s =< x_0, y_0, y_1 >$. *Then, for example,*

$$\lambda\left(s, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

If, in the definition of finite transducer with memory, $(\mathcal{Y}, +)$ is a group (not necessarily Abelian) and the function ϕ is of the form

$$\phi = f(x_h, x_{h-1}, \ldots, x_1, x_0) + g(y_{k-1}, \ldots, y_1, y_0),$$

for some $f : \mathcal{X}^{h+1} \to \mathcal{Y}$ and $g : \mathcal{Y}^k \to \mathcal{Y}$, one says that M_ϕ is a *separable finite transducer with memory*, denoted by $M_{f,g}$. Notice that, in particular, a finite transducer with input-only-memory is a separable finite transducer.

Example 3.4.4. *The transducer defined in the previous example is a separable finite transducer, while the transducer presented in* Example *3.4.2 is not separable.*

Theorem 3.4.5. *Let* $\mathcal{Y}$ *be a group, denoted additively. Then, the separable finite transducer* $M_{f,g} = \left\langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta_{f,g}, \lambda_{f,g} \right\rangle$ *is injective with delay* τ *if and only if the transducer* $M_f = \left\langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h, \delta_f, \lambda_f \right\rangle$ *is injective with delay* τ.

Proof. Notice that, given $s_x \in \mathcal{X}^h$, $s_y \in \mathcal{Y}^k$, $x \in \mathcal{X}$, one can write

$$\lambda_{f,g}(< s_x, s_y >, x) = f(s_x, x) + g(s_y).$$

Also, if $\alpha \in \mathcal{X}^\tau$, then $\lambda_{f,g}(< s_x, s_y >, x\alpha)$ is just a sequence of elements as in the previous equation.

Since, obviously, for all $x, x' \in \mathcal{X}$

$$f(s_x, x) + g(s_y) = f(s_x, x') + g(s_y) \iff f(s_x, x) = f(s_x, x'),$$

one concludes that, for $\alpha \in \mathcal{X}^\tau$,

$$\lambda_{f,g}(< s_x, s_y >, x\alpha) = \lambda_{f,g}(< s_x, s_y >, x'\alpha') \iff \lambda_f(< s_x >, x\alpha) = \lambda_f(< s_x >, x'\alpha').$$

From this, the claim made follows immediately. □

Let $M = \langle \mathcal{X}, \mathcal{Y}, S, \delta, \lambda \rangle$ be a finite transducer. In *Section* 3.2, it was proved that if M is ω-injective, then there exists a non-negative integer $\tau \leq \frac{|S|(|S|-1)}{2}$ such that M is injective with delay τ (*Corollary* 3.2.9). Thus, to check if M is ω-injective, in the worst case, one has to verify if M is τ-injectivity for $\tau = 0, 1, 2, \ldots, \frac{|S|(|S|-1)}{2}$. For example, let $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^2, (\mathbb{F}_2^2)^3, \delta, \lambda \right\rangle$ be a finite transducer with input memory 3. Since $|S| = |(\mathbb{F}_2^2)^3| = 64$, to check if M is injective, in the worst case, one has to verify if M is τ-injective for $\tau = 0, 1, \ldots, 2016$. If the transducer has input memory of order 4, then $|S| = |(\mathbb{F}_2^2)^4| = 256$, and the number of checkings rises to 32640. It is easy to see that the number of verifications grows exponentially.

Considering the special structure of finite transducers with memory, it is plausible that the number of checks required is lower. After some practical tests checking the injectivity of finite transducers with memory and some partial ideas for a proof (see *Appendix* A), we suspect that a finite transducer with memory $M = \left\langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \right\rangle$ is ω-injective if and only if there exists a non-negative integer $\tau \leq h \dim(\mathcal{X})$ such that M is injective with delay τ.

3.4.1 Linear Finite Transducers with Memory

Linear finite transducers with memory, as the name suggests, are associated with linear functions. In agreement with *Definition* 3.4.1, a *linear finite transducer with memory* is completely defined by its memory and by the linear function ϕ.

Example 3.4.6. *The transducer defined in* Example *3.4.3 is a* linear finite transducer with memory*, while the transducer presented in* Example *3.4.2 is not linear.*

Example 3.4.7. *Let* $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^2, (\mathbb{F}_2^2)^3 \times (\mathbb{F}_2^3)^2, \delta, \lambda \right\rangle$ *be the finite transducer with memory of order* $(3,2)$ *defined by*

$$y_{t+2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} x_t + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} y_t, \text{ for } t \geq 0,$$

where $x_t \in \mathbb{F}_2^2$*,* $y_t \in \mathbb{F}_2^2$*, for* $t \geq 0$*. This transducer is a* linear finite transducer with memory.

Let M be a linear finite transducer with memory (h,k) defined by $\phi : \mathcal{X}^{h+1} \times \mathcal{Y}^k \longrightarrow \mathcal{Y}$, with $h, k \in \mathbb{N}_0$ and $(\mathcal{Y}, +)$ a group. Since ϕ is a linear function, one can always separate it in two function, $f : \mathcal{X}^{h+1} \to \mathcal{Y}$ and $g : \mathcal{Y}^k \to \mathcal{Y}$, such that $\phi(x_h, \ldots, x_0, y_{k-1}, \ldots, y_0) = f(x_h, \ldots, x_0) + g(y_{k-1}, \ldots, y_0)$. In this way, all linear finite transducers are separable. From *Theorem* 3.4.5, one can conclude that the study of injectivity of linear finite transducers with memory can be reduced to the study of LFTs with input-only-memory.

Linear finite transducers with memory can be defined as an infinite system of linear equations that relates the sequence of inputs and outputs. Let $\mathcal{X}$ be the input alphabet, $\mathcal{Y}$ the output alphabet and $h, k \in \mathbb{N}_0$. Let $\mathbb{S}_{h,k}$ be the set of infinite systems, in the variables $(x_t)_{t\geq 0} \in \mathcal{X}$, $(y_t)_{t\geq 0} \in \mathcal{Y}$, of the form:

$$\sum_{j=0}^{h} A_j \, x_{t+h-j} + \sum_{j=0}^{k+r} B_j \, y_{t+j} = 0, \text{ for } t \geq 0,$$

where $r = h \dim(\mathcal{X})$. Any infinite system that defines a linear transducer with memory of order (h,k) can be seen as a system in $\mathbb{S}_{h,k}$, if one considers as many null matrices as necessary to "complete" the general equation.

Example 3.4.8. *Let* $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \right\rangle$ *be the linear finite transducer with memory of order* $(2,0)$ *defined by the infinite system*

$$y_t = A_0 x_{t+2} + A_1 x_{t+1} + A_2 x_t = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

This transducer is the following system in $\mathbb{S}_{2,0}$*:*

$$\sum_{j=0}^{h} A_j\, x_{t+h-j} + \sum_{j=0}^{k+r} B_j\, y_{t+j} = 0 \iff \sum_{j=0}^{2} A_j\, x_{t+2-j} + \sum_{j=0}^{0+2\times 3} B_j\, y_{t+j} = 0$$
$$\iff \sum_{j=0}^{2} A_j\, x_{t+2-j} + \sum_{j=0}^{6} B_j\, y_{t+j} = 0, \text{ for } t \geq 0,$$

where A_j*, for* $j = 0, 1, 2$*, are as before,* $B_0 = I$ *and* $B_j = 0$*, for* $j = 1, 2, \ldots, 6$*.*

Now, we define two concepts over these systems that will be very useful to present, later in this work, a necessary and sufficient condition for the injectivity of linear finite transducers with memory.

Definition 3.4.9. *Let* $h, k \in \mathbb{N}_0$ *and let* S *be a system in* $\mathbb{S}_{h,k}$*. The rank of* S *is the rank of the coefficient matrix of* x_{t+h} *and is denoted by* $rank(S)$*, that is,* $rank(S) = rank(A_0)$*.*

Example 3.4.10. *Consider the transducer presented in* Example *3.4.8 and the infinite system* S *associated. Then,*

$$rank(A_0) = rank\left(\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) = 1 \;=\; rank(S).$$

Definition 3.4.11. *A system* S *in* $\mathbb{S}_{h,k}$ *is in reduced form if the first* $rank(S)$ *rows of* A_0 *are linearly independent and the other ones are null.*

Notice that the reduced form of a system is not unique.

Example 3.4.12. *A reduced form of the transducer* $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \rangle$ *with input memory* 2 *presented in* Example *3.4.8 defined by the infinite system*

$$y_t = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0,$$

is obtained by adding the first row to the second one:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

3.4.2 Quasi-Linear Finite Transducers with Memory

Quasi-linear finite transducers (QLFT) with memory were introduced, as far as we know, by Renji Tao [Tao09]. The main idea behind the concept of QLFTs is to introduce some kind of non-linearity to the definition of linear finite transducers. The linear and quasi-linear finite transducers with memory are easy to invert. In the composition of these two types of transducers, the non-linear part blends with the linear part resulting in a non-linear transducer. It is not known a procedure to invert non-linear transducers, therefore, one can only invert a non-linear transducer by knowing the original factors, in this case the linear and quasi-linear transducers.

In his book, Tao defined *τ-quasi-linear finite transducers* forcing the most recent $\tau + 1$ input symbols to only appearing in the linear part of the transducer. In this way, he was able to "extend" the known results about linear finite transducers injectivity.

Definition 3.4.13. *Let $h, k \in \mathbb{N}_0$ and $\tau \in \mathbb{N}_0$ such that $\tau \leq h$. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a finite transducer with memory of order (h, k). If M is defined by an equation of the form*

$$y_{t+k} = \sum_{j=0}^{\tau} A_j x_{t+h-j} + f(x_t, x_{t+1}, \ldots, x_{t+h-\tau-1}, y_t, y_{t+1}, \ldots, y_{t+(k-1)}), \text{ for } t \geq 0,$$

where $f : \mathcal{X}^{h-\tau} \times \mathcal{Y}^k \to \mathcal{Y}$ is a non-linear function, then we say that M is a τ-quasi-linear finite transducer (τ-QLFT).

Example 3.4.14. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{12}, \delta, \lambda \rangle$ be the finite transducer with input memory of order 4 defined by*

$$\begin{aligned} y_t &= A_0 x_{t+4} + A_1 x_{t+3} + A_2 x_{t+2} + f(x_t, x_{t+1}) \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} x_{t+2} + f(x_t, x_{t+1}), \text{ for } t \geq 0, \end{aligned}$$

where $(x_t)_{t \geq 0} \in \mathbb{F}_2^3$, $s =< x_0, x_1, x_2, x_3 > \in \mathbb{F}_2^{12}$ is the initial state of the transducer and f is a non-linear function (for example, componentwise multiplication). Then, one can say that M is a 2-quasi-linear finite transducer.

In what follows, it is proved that the problem of checking injectivity of τ-QLFTs can be reduced to the problem of checking injectivity of linear finite transducers.

Let $h, k \in \mathbb{N}_0$ and $\tau \in \mathbb{N}_0$ such that $\tau \leq h$. Let $f : \mathcal{X}^{h-\tau} \times \mathcal{Y}^k \to \mathcal{Y}$ be a non-linear function. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a τ-QLFT defined by

$$y_{t+k} = \sum_{j=0}^{\tau} A_j x_{t+h-j} + f(x_t, x_{t+1}, \ldots, x_{t+h-\tau-1}, y_t, y_{t+1}, \ldots, y_{t+(k-1)}), \text{ for } t \geq 0.$$

Now, one has to construct an LFT from M as follows. Let $M_L = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^\tau, \delta_L, \lambda_L \rangle$ be the linear finite transducer with memory of order $(\tau, 0)$ defined by

$$y_t = \sum_{j=0}^{\tau} A_j x_{t+\tau-j}, \text{ for } t \geq 0.$$

Basically, to construct M_L, we dropped the non-linear part of M.

Theorem 3.4.15. *Let $r \in \mathbb{N}_0$ such that $r \leq \tau$. M is invertible with delay r if and only if M_L is invertible with delay r.*

Proof. Since, $\forall s =< x_0, \ldots, x_{h-1}, y_0, \ldots, y_{k-1} > \in \mathcal{X}^h \times \mathcal{Y}^k$, $\forall x, x' \in \mathcal{X}$, one has:

$$\lambda(s, x) = \lambda(s, x')$$
$$\lambda_L(s_L, x) + f(x_0, \ldots, x_{h-\tau-1}, y_0, \ldots y_{k-1}) = \lambda_L(s_L, x') + f(x_0, \ldots, x_{h-\tau-1}, y_0, \ldots y_{k-1})$$
$$\lambda_L(s_L, x) = \lambda_L(s_L, x')$$

where $s_L =< x_{h-\tau}, \ldots, x_{h-1} >$. Then, $\forall x, x' \in \mathcal{X}$, $\forall \alpha, \alpha' \in \mathcal{X}^r$ one has

$$\lambda(s, x\alpha) = \lambda(s, x'\alpha') \iff \lambda_L(s_L, x\alpha) = \lambda_L(s_L, x'\alpha'),$$

then, M is invertible with delay r if and only if M_L is invertible with delay r. □

Although Tao only defined and studied τ-QLFT and its injectivity, let us define a general quasi-linear finite transducer.

Definition 3.4.16. *Let $h, k \in \mathbb{N}_0$. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a finite transducer with memory of order (h, k). M is a quasi-linear finite transducer if it is defined by an equation of the form*

$$y_{t+k} = \sum_{j=0}^{h} A_j x_{t+h-j} + \sum_{j=0}^{k-1} B_j y_{t+j} + f(x_t, x_{t+1}, \ldots, x_{t+(h-1)}, y_t, y_{t+1}, \ldots, y_{t+(k-1)}), \text{ for } t \geq 0,$$

where $f : \mathcal{X}^h \times \mathcal{Y}^r \to \mathcal{Y}$ is a non-linear function.

Basically, in this way, one can take any linear finite transducer and transform it in a quasi-linear finite transducer by adding a non-linear function. This fact, by itself, allows us to expand the space of possible private keys.

Example 3.4.17. *Let* $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^9, \delta, \lambda \right\rangle$ *be the finite transducer with input memory of order* 3 *defined by:*

$$\begin{aligned} y_t &= A_0 x_{t+3} + A_1 x_{t+2} + A_2 x_{t+1} + A_3 x_t + f(x_t, x_{t+1}, x_{t+2}) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} \cdot x_t, \end{aligned}$$

for $t \geq 0$*, where* $(x_t)_{t \geq 0} \in \mathbb{F}_2^3$ *and* $s_0 =< x_0, x_1, x_2 > \in \mathbb{F}_2^3$ *is the initial state of the transducer. This transducer is a quasi-linear finite transducer, only in the general sense.*

Later, in this work, it will be presented a necessary and sufficient condition for these transducers to be injective.

Chapter 4

Invertibility of Finite Transducers with Memory

In the last chapter, it was introduced the different types of transducers as well as the definitions of ω-injective transducer and injective transducer with delay τ, $\tau \in \mathbb{N}_0$. Furthermore, it was presented several results related to the problem of checking finite transducers injectivity. This suggests that the transducers injectivity is an important fact.

In all cryptosystems, the security relies on the difficulty of being able to invert the encryption procedure without knowing the private key. As usual in a public key cryptosystem, the encryption is done using the public key, which in the case of FAPKC is a non-linear finite transducer that is the composition of the private key transducers (one linear and one quasi-linear). To invert the encryption procedure, one has to compute the inverse of the public key transducer. As said before, it is not known a method to invert non-linear finite transducers. However, it is possible to decrypt a ciphertext if one finds inverses of the transducers in the private key (which only the owner knows) and compute their composition. This compound transducer is the inverse of the public key transducer used in the encryption process.

In this chapter, it will be presented a procedure to check if a linear transducer with memory is injective, as well as a procedure to compute its inverse, if it exists. In fact, during the procedure of checking injectivity, it is constructed the inverse of the transducer. Also, these results will be extended to quasi-linear finite transducers.

4.1 Criterium of Invertibility of LFTs with Memory

The method proposed by Renji Tao to check the invertibility of a linear finite transducer uses a pair of transformations that he calls R_a and R_b transformations, applied to the equations of the infinite system that defines the transducer [Tao09]. Roughly speaking, R_a transformations are used to obtain an equivalent system in reduced form, and R_b transformations are used to discard irrelevant equations of the infinite system and reorganize the others to obtain a new system in the same form.

In what follows, the R_a and R_b transformations will be formalized as well as the procedure to check τ-injectivity on linear finite transducers with memory, for $\tau \in \mathbb{N}_0$. Additionally, this procedure will be illustrated through an example.

To formalise the procedure that checks linear finite transducers τ-injectivity, let us start by introducing two auxiliary matrices that will be used in the R_b transformations applied to the infinite system (as will be introduced in the next definition). Let $c, n \in \mathbb{N}$ where $c \leq n$,

$$I^{+}_{c,n} = \left[\begin{array}{c:c} I_c & 0_{c\times(n-c)} \\ \hdashline 0_{(n-c)\times c} & 0_{n-c} \end{array}\right] \quad \text{and} \quad I^{-}_{c,n} = \left[\begin{array}{c:c} 0_{n-c} & 0_{(n-c)\times c} \\ \hdashline 0_{c\times(n-c)} & I_c \end{array}\right].$$

Basically, this matrices will allow us to "merge" the information needed from two matrices in one matrix, as can be seen in the next example.

Example 4.1.1. *Consider the following matrices:*

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} \quad \textit{and} \quad B = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix}.$$

To produce a matrix that has the first row of A (and all other rows null) one can use $I^{+}_{1,3}$ in the following way:

$$I^{+}_{1,3}A = \left[\begin{array}{c:cc} 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

If one wants a matrix with the last 2 rows of B then it is useful to use $I^{-}_{2,3}$*:*

$$I^{-}_{2,3}B = \left[\begin{array}{c:cc} 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix}.$$

To construct a matrix with the first row of A and the last 2 rows of B one can proceed as follows:

$$I^{+}_{1,3}A + I^{-}_{2,3}B = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ b_{1,0} & b_{1,1} & b_{1,2} \\ b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix}.$$

Let M be a linear finite transducer with memory of order (h, k), where $h, k \in \mathbb{N}_0$, defined by an infinite system of linear equations $\mathbb{S}_{h,k}$ (as presented in *Section* 3.4.1). The procedure used to verify if M is injective is formalized here as a transformation $G_{h,k}$ mapping the set $\mathbb{S}_{h,k}$ to itself.

Definition 4.1.2. *Let M be a transducer as before and $n \in \mathbb{N}$ be the number of rows of the matrix A_0. Let $G_{h,k} : \mathbb{S}_{h,k} \to \mathbb{S}_{h,k}$ be the transformation that assigns to each system $S \in \mathbb{S}_{h,k}$ the system in $\mathbb{S}_{h,k}$ obtained in the following way:*

R_a **transformation:** *applies to S a sequence of elementary row operations to obtain an equivalent system in reduced form;*

R_b **transformation:** *discards the equations of the system obtained above that do not depend on x_{t+h} or other subsequent inputs, i.e, the ones corresponding to the $n - rank(S)$ null rows of A_0, and reorganize the others by putting together the equations that depend on x_{t+h} and do not depend on subsequent inputs, for $t \geq 0$ (this is easier to understand with an example – see* Example *4.1.4).*

It is easy to see that, when the matrix A_0 has full rank, a $G_{h,k}$ transformation is reduced to a R_a transformation and it is obtained an equivalent system.

Remark 4.1.3. *The map $G_{h,k}$ is indeed well-defined, i.e, its images are in $\mathbb{S}_{h,k}$, as can be seen as follows. Let $h, k \in \mathbb{N}_0$ and let S be the system in $\mathbb{S}_{h,k}$ defined by:*

$$\sum_{j=0}^{h} A_j^{(0)}\, x_{t+h-j} + \sum_{j=0}^{k+r} B_j^{(0)}\, y_{t+j} = 0, \text{ for } t \geq 0 \text{ and } r = h\dim(x_t).$$

Now, let

$$\sum_{j=0}^{h} \tilde{A}_j^{(0)}\, x_{t+h-j} + \sum_{j=0}^{k+r} \tilde{B}_j^{(0)}\, y_{t+j} = 0, \text{ for } t \geq 0,$$

be the system obtained after the first step of determining $G_{h,k}(S)$, this is, after applying the R_a transformation. Then, $G_{h,k}(S)$ is the system

$$\sum_{j=0}^{h} A_j^{(1)}\, x_{t+h-j} + \sum_{j=0}^{k+r} B_j^{(1)}\, y_{t+j} = 0, \text{ for } t \geq 0,$$

where

$$A_j^{(1)} = I_{c,m}^{+} \tilde{A}_j^{(0)} + I_{m-c,m}^{-} \tilde{A}_{j+1}^{(0)},$$
$$B_j^{(1)} = I_{c,m}^{+} \tilde{B}_j^{(0)} + I_{m-c,m}^{-} \tilde{B}_{j+1}^{(0)},$$

$c = rank(S)$ *and* $A_{h+1}^{(0)} = B_{k+r+1}^{(0)} = 0$. *One concludes that* $G_{h,k}(S) \in \mathbb{S}_{h,k}$.

Let $h, k \in \mathbb{N}_0$ and take $S \in \mathbb{S}_{h,k}$. Since $G_{h,k}$ is a transformation in $\mathbb{S}_{h,k}$, one can define $G_{h,k}^{\tau}$ as the τ-th iterate of $G_{h,k}$, where $\tau \in \mathbb{N}_0$, by

$$G_{h,k}^{0} = id_{\mathbb{S}_{h,k}}$$
$$G_{h,k}^{\tau+1} = G_{h,k}^{\tau} \circ G_{h,k}(S)$$

where $id_{\mathbb{S}_{h,k}}$ is the identity transformation on $\mathbb{S}_{h,k}$ and $G_{h,k}^{\tau} \circ G_{h,k}(S) = G_{h,k}(G_{h,k}(S))$.

In the next example, it will be illustrated how $G_{h,k}$ transformations can be used to check if a linear finite transducer with memory $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ is invertible. What is done in the example is to verify if there exists $\tau \in \mathbb{N}_0$ such that $\forall s \in S$, $\forall x \in \mathcal{X}$, $\forall \alpha \in \mathcal{X}^{\tau}$, x is uniquely determined by s and $\lambda(s, x\alpha)$.

Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h, \delta, \lambda \rangle$ be a linear finite transducer with input memory of order h. To illustrate the procedure, it will be used a transducer with input-only-memory, because, besides being simpler, the problem of testing injectivity of linear transducers with memory can be reduced to the problem of checking injectivity of linear transducers with input-only-memory (*Section* 3.4.1). Furthermore, recall that, since M is a transducer with input memory h, if M is τ-injective then $\tau \leq h \dim(\mathcal{X})$.

Let M be a transducer as before. Let $x_h x_{h+1} x_{h+2} \ldots$ be an input sequence, where $(x_t)_{t \geq h} \in \mathcal{X}$, and $s =< x_0, x_1, \ldots, x_{h-1} >\in \mathcal{X}^h$ be the initial state of the transducer. Then, the output

sequence of the transducer is given by $y_0y_1y_2\ldots = \lambda(s, x_hx_{h+1}x_{h+2}\ldots)$, where $(y_t)_{t\geq 0} \in \mathcal{Y}$. In the following example, to check if the transducer M is injective, i.e., if it is possible to recover the first input symbol, it will be proceed sequentially as follows:

- First, one checks if, given s and y_0, x_h is uniquely determined by them. If that is true, the transducer is invertible with delay 0;
- Otherwise, one checks if x_h is uniquely determined by s, y_0 and y_1. In that case the transducer is invertible with delay 1;
- This can be continued until one checks if x_h is uniquely determined by s and $y_0, y_1, \ldots,$ $y_{h\dim(\mathcal{X})}$. If that is true the transducer is invertible with delay $h\dim(\mathcal{X})$ and x_h can be recovered. Otherwise, one can conclude that the transducer is not injective.

In fact, checking if x_h is uniquely determined by s and $y_0, y_1, \ldots, y_\tau$ is equivalent to check if the matrix A_0 of the infinite system in the τ-th $G_{h,k}$ transformation has full rank.

Notice that, after reaching a system where the matrix A_0 has full rank, if one continues to apply $G_{h,k}$ transformations, it will be obtained equivalent systems, therefore the procedure ends when the matrix A_0 has full rank. This means that the transducer M is injective with delay equals to the number of $G_{h,k}$ transformations applied, or when the limit of the τ-injectivity is reached.

Example 4.1.4. *Let $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \right\rangle$ be the linear finite transducer with input memory of order 2 defined by the infinite system*

$$\begin{aligned} y_t &= A_0x_{t+2} + A_1x_{t+1} + A_2x_t \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0, \end{aligned}$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$ and $s_0 =< x_0, x_1 >$ is the initial state of the transducer. Let $(x_t)_{t\geq 2} \in \mathbb{F}_2^3$ be an input sequence and consider $(y_t)_{t\geq 0} = \lambda(s_0, (x_t)_{t\geq 2})$.

Notice that, the second and the third columns of A_0 are null. Therefore, y_0 does not contain any information about the second and the third components of x_2. Consequently, x_2 is not uniquely determined by y_0 and s_0, i.e, the transducer is not invertible *with delay* 0.

Adding the knowledge of y_1, by the procedure presented before, allows to uniquely determine x_2. The first step is to apply to the system a sequence of elementary row operations to obtain an

equivalent system in reduced form, that is, to apply a R_a transformation. This can be obtained (as was done in Example *3.4.12) by adding the first row to the second one:*

$$\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_{t+1} + \begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_t,\ for\, t \geq 0.$$

Then, let us expand the new system in the following way:

$$\begin{cases}
\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} y_0 = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_2 + \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_1 + \begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_0 \\
\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} y_1 = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_3 + \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_2 + \begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_1 \\
\vdots \\
\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_{t+1} + \begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_t \\
\begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} y_{t+1} = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_{t+3} + \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_{t+1} \\
\vdots
\end{cases}.$$

There are 2 equations $(3 - rank(S))$ that do not depend on x_2 and therefore can be discarded for the purpose of obtaining x_2 from s_0 and y_0y_1. The next step is to discard those equations and to reorganize the others by putting together the equations that depend on x_{t+2} and do not depend on subsequent inputs, this is, applying a R_b transformation. To do that, for any two consecutive matricial equations, reallocate the rows by putting together the first 1 $(rank(S))$ row of the first equation and the last 2 $(3 - rank(S))$ rows of the second equation. The result

of this procedure is the following system:

$$\begin{bmatrix}0&0&0\\1&1&0\\0&0&1\end{bmatrix} y_{t+1}+\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix} x_{t+2}+\begin{bmatrix}1&0&0\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}+\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} x_t, \text{ for } t \geq 0.$$

The coefficient matrix of x_{t+2} is invertible, i.e, the matrix A_0 of $G_{2,0}(S)$ has full rank, therefore, using this new system, x_2 is uniquely determined by s_0, y_0 and y_1, that is, the transducer is injective with delay 1.

4.2 Inverses of LFTs with memory

The previous section included a necessary and sufficient condition for a linear finite transducer $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$, with memory of order (h,k), $h,k \in \mathbb{N}_0$, defined by a system $S \in \mathbb{S}_{h,k}$, to be invertible with delay $\tau \leq h \dim(\mathcal{X})$, being that condition the matrix A_0 in $G^{\tau}_{h,k}(S)$ to have maximum rank.

In this section, it will be shown how to construct an inverse of linear finite transducers with memory of order (h,k), in case of having one. In order to be able to demonstrate the results associated with the inverses, first let us introduce the following lemmas.

Lemma 4.2.1. *Let $h,k \in \mathbb{N}_0$ and let S be a system in $\mathbb{S}_{h,k}$, where the matrix A_0 has n rows, $n \in \mathbb{N}$. Then:*

P1 *S implies $G_{h,k}(S)$, i.e., if $(x_t, y_t)_{t\geq 0}$ is a solution of S, then it is also a solution of $G_{h,k}(S)$;*

P2 *if $(x_t, y_t)_{t\geq 0}$ is a solution of $G_{h,k}(S)$ then $(x_t, y_t)_{t\geq 1}$ is a solution of S.*

Proof. The property *P1* is quite obvious since $G_{h,k}(S)$ is obtained from S though a sequence of elementary row operations, which preserve systems equivalence, and then some equations of the system are removed. Thus, the set of solutions of S is a subset of the solutions of $G_{h,k}(S)$.

To prove the second property notice that $G_{h,k}(S)$ is obtained from a system equivalent to S by removing a subset of its first n equations. Then, if $(x_t, y_t)_{t\geq 0}$ is a solution of $G_{h,k}(S)$, it is also a solution of the system obtained by removing all the first n equations, which is here denoted by $G_{h,k}(S)^*$. S is a system defined by $\sum_{j=0}^{h} A_j\, x_{t+h-j} + \sum_{j=0}^{k+r} B_j\, y_{t+j} = 0$, for $t \geq 0$, or equivalently,

$$S: \begin{bmatrix} A_h & A_{h-1} & A_{h-2} & \dots & A_0 & 0 & 0 & 0 & \dots \\ 0 & A_h & A_{h-1} & \dots & A_1 & A_0 & 0 & 0 & \dots \\ 0 & 0 & A_h & \dots & A_2 & A_1 & A_0 & 0 & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} +$$

$$\begin{bmatrix} B_0 & B_1 & B_2 & \dots & B_{k+r} & 0 & 0 & 0 & \dots \\ 0 & B_0 & B_1 & \dots & B_{k+r-1} & B_{k+r} & 0 & 0 & \dots \\ 0 & 0 & B_0 & \dots & B_{k+r-2} & B_{k+r-1} & B_{k+r} & 0 & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix}.$$

The system $G_{h,k}(S)^*$, obtained from S by removing the first n equations, is defined by:

$$G(S)^*: \begin{bmatrix} 0 & A_h & A_{h-1} & \dots & A_1 & A_0 & 0 & 0 & \dots \\ 0 & 0 & A_h & \dots & A_2 & A_1 & A_0 & 0 & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} +$$

$$\begin{bmatrix} 0 & B_0 & B_1 & \dots & B_{k+r-1} & B_{k+r} & 0 & 0 & \dots \\ 0 & 0 & B_0 & \dots & B_{k+r-2} & B_{k+r-1} & B_{k+r} & 0 & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

or, $0x_0 + \sum_{j=0}^{h} A_j\, x_{t+h-j} + 0y_0 + \sum_{j=0}^{k+r} B_j\, y_{t+j} = 0$, for $t \geq 1$. Lets assume that $(x_t, y_t)_{t\geq 0}$ is a solution of $G_{h,k}(S)$ (also a solution of $G_{h,k}(S)^*$), then:

$$\sum_{j=0}^{h} A_j\, x_{t+h-j} + \sum_{j=0}^{k+r} B_j\, y_{t+j} = 0,\ t \geq 1 \iff \sum_{j=0}^{h} A_j\, x_{t+1+h-j} + \sum_{j=0}^{k+r} B_j\, y_{t+1+j} = 0,\ t \geq 0$$

$$\iff (x_{t+1}, y_{t+1})_{t\geq 0} = (x_t, y_t)_{t\geq 1} \text{ is a solution of } S.$$

□

This properties can naturally be extended to $G^{\tau}_{h,k}$, for $\tau \geq 0$, in the following way:

Lemma 4.2.2. *Let $h, k \in \mathbb{N}_0$ and let S be the system in $\mathbb{S}_{h,k}$. Then:*

P1 *S implies $G^{\tau}_{h,k}(S)$, i.e., if $(x_t, y_t)_{t\geq 0}$ is a solution of S, then it is also a solution of $G^{\tau}_{h,k}(S)$;*

P2 *if $(x_t, y_t)_{t\geq 0}$ is a solution of $G^{\tau}_{h,k}(S)$ then $(x_t, y_t)_{t\geq \tau}$ is a solution of S.*

In the next result it is shown a way to obtain the inverse of a τ-injective linear finite transducer with memory of order $(h,0)$.

Theorem 4.2.3. *Let $\ell \in \mathbb{N}$. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h, \delta, \lambda \rangle = \langle \mathbb{F}_2^\ell, \mathbb{F}_2^\ell, \mathbb{F}_2^{h\ell}, \delta, \lambda \rangle$ be the linear finite transducer with memory of order $(h,0)$ defined by the infinite system:*

$$S : \sum_{j=0}^{h} A_j^{(0)}\, x_{t+h-j} + \sum_{j=0}^{r} B_j^{(0)}\, y_{t+j} = 0, \text{ for } t \geq 0,$$

where $r = h\ell$, $A_j^{(0)} \in \mathcal{M}_\ell$, $B_0^{(0)} = I_\ell$ and $B_j^{(0)} = 0_\ell$ for $0 < j \leq r$. If $G_{h,0}^{\tau}(S)$ is the system

$$\sum_{j=0}^{h} A_j^{(\tau)}\, x_{t+h-j} + \sum_{j=0}^{r} B_j^{(\tau)}\, y_{t+j} = 0, \text{ for } t \geq 0,$$

then M is invertible with delay τ if and only if $A_0^{(\tau)}$ is an invertible matrix. Let L be the inverse matrix of $A_0^{(\tau)}$. Let $M^* = \langle \mathcal{Y}, \mathcal{X}, \mathcal{Y}^\tau \times \mathcal{X}^h, \delta^*, \lambda^* \rangle = \left\langle \mathbb{F}_2^\ell, \mathbb{F}_2^\ell, \mathbb{F}_2^{(\tau+h)\ell}, \delta^*, \lambda^* \right\rangle$ be the linear finite transducer with memory of order (τ, h) obtained by multiplying, on the left, both sides of $G_{h,0}^{\tau}(S)$ by L:

$$\sum_{j=0}^{h} LA_j^{(\tau)}\, x_{t+h-j} + \sum_{j=0}^{r} LB_j^{(\tau)}\, y_{t+j} = 0, \text{ for } t \geq 0.$$

Then, M^* is a left inverse with delay τ of M, and M is a left inverse with delay τ of M^*.

Proof. First, notice that, in the transducer M^*, $LA_0^{(\tau)}$ is the identity matrix and $B_j^{(\tau)} = 0$ for $j > \tau$. To construct M^*, the inverse of M, one has to see the transducer "in reverse", that is, see the input as output and vice versa. Since M is injective with delay τ, it will need the information of $(y_i)_{0 \leq i \leq \tau}$ to invert the transducer, therefore, the inverse will have input memory of order τ. Also, the inverse transducer will have output memory h, the input memory of M.

To prove that M^* is a left inverse with delay τ of M, one has to demonstrate that $\forall s \in \mathcal{X}^h, \exists s^* \in \mathcal{Y}^\tau \times \mathcal{X}^h : \forall \alpha \in \mathcal{X}^\omega,\ \lambda^*(s^*, \lambda(s,\alpha)) = \gamma\alpha$ for some $\gamma \in \mathcal{X}^\tau$ (by *Definition* 3.2.11 and *Definition* 3.2.14). Let $s =< x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1} >$ be a generic state of M. It will be proved that any state s^* of M^* of the form $< y_0, y_1, \ldots, y_{\tau-1}, x_0, x_1, \ldots, x_{h-1} >$ inverts s with delay τ, where $x_0, x_1, \ldots, x_{\tau-1}$ are arbitrary elements in $\mathcal{X}$ (notice that the values of $x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1}$ are in s) and

$$y_0 y_1 \ldots y_{\tau-1} = \lambda(< x_0, x_1, \ldots, x_{h-1} >, x_h x_{h+1} \ldots x_{h+\tau-1}).$$

Now, consider the input sequence $(x_t)_{t \geq h+\tau}$ and

$$(y_t)_{t \geq \tau} = \lambda(< x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1} >, (x_t)_{t \geq h+\tau}).$$

From the last two equations, one has $(y_t)_{t\geq 0} = \lambda(< x_0, x_1, \ldots, x_{h-1} >, (x_t)_{t\geq h})$. This means that $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 0}$ satisfy the system of equations S that defines de transducer M. Then, from *Property P1* of *Lemma* 4.2.2, $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 0}$ also satisfy the system $G^{\tau}_{h,0}(S)$. Consequently, since the system that defines the transducer M^* is obtained from $G^{\tau}_{h,0}(S)$ by multiplying this by the matrix L, $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 0}$ also satisfy the system of M^*, that is,

$$(x_t)_{t\geq h} = \lambda^*(< y_0, y_1, \ldots, y_{\tau-1}, x_0, x_1, \ldots, x_{h-1} >, (y_t)_{t\geq \tau}) \iff$$
$$(x_t)_{t\geq h} = \lambda^*(< y_0, y_1, \ldots, y_{\tau-1}, x_0, x_1, \ldots, x_{h-1} >, \lambda(< x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1} >, (x_t)_{t\geq h+\tau})).$$

Then, it was proved that, for $\alpha = (x_t)_{t\geq h+\tau} \in \mathcal{X}^\omega$, $\lambda^*(s^*, \lambda(s, \alpha)) = \gamma\alpha$ where $\gamma = (x_t)_{h\leq t<h+\tau} \in \mathcal{X}^\tau$. Therefore, M^* is a left inverse with delay τ of M.

To prove the second claim that M is a left inverse with delay τ of M^*, let $s^* =< y_0, y_1, \ldots, y_{\tau-1}, x_0, x_1, \ldots, x_{h-1} >$ be a generic state of M^* and $s =< x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1} >$ a state of S, where $x_h, x_{h+1} \ldots, x_{\tau+h-1}$ are arbitrary elements in $\mathcal{X}$ (notice that the symbols $x_\tau, x_{\tau+1}, \ldots, x_{h-1}$ are in s^*). It will be demonstrated that s inverts s^* with delay τ.

Consider the input sequence $(y_t)_{t\geq\tau} \in \mathcal{Y}^\omega$, then

$$(x_t)_{t\geq h} = \lambda^*(< y_0, \ldots, y_{\tau-1}, x_0, \ldots, x_{h-1} >, (y_t)_{t\geq\tau}).$$

This means that $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 0}$ satisfy the infinite system that defines the transducer M^* and, consequently, satisfy the system $G^{\tau}_{h,0}(S)$. Then, from *Property P2* of *Lemma* 4.2.2, $(x_t)_{t\geq\tau}$ and $(y_t)_{t\geq\tau}$ satisfy the system associated to the transducer M, i.e,

$$\begin{aligned}(y_t)_{t\geq\tau} &= \lambda(< x_\tau, x_{\tau+1}, \ldots, x_{\tau+h-1} >, (x_t)_{t\geq\tau+h}) \\ &= \lambda(s, (x_t)_{t\geq\tau+h}).\end{aligned}$$

This way, one can recover the input sequence $(y_t)_{t\geq\tau} \in \mathcal{Y}^\omega$ so, s is an inverse state of s^*. Futhermore, M is a left inverse with delay τ of M^*. □

Although the last result only shows the construction of inverses of linear finite transducers with memory of order $(h, 0)$, the result can be extended to transducers with memory (h, k). In fact, the formula to the inverse transducer is the same as presented before, and the proof is analogue.

As seen in the last theorem, the inverse of a linear finite transducer is constructed during the procedure of checking its injectivity.

Example 4.2.4. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \rangle$ be the infinite transducer with memory $(2,0)$ defined by the infinite system*

$$y_t = A_0x_{t+2} + A_1x_{t+1} + A_2x_t = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0,$$

where $x_t, x_{t+1}, x_{t+2} \in \mathbb{F}_2^3$ and $s_0 =< x_0, x_1 >$ is the initial state of the transducer.

In Example *4.1.4 was computed the infinite system $G_{2,0}(S)$ which has full rank:*

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_t = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

Therefore, M is invertible. To obtain the inverse transducer of M one only has to multiply, on the left, the previous equation by the inverse matrix of A_0. Since

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

one can obtain the inverse transducer by multiplying the infinite system $G_{2,0}(S)$ by $L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

Then, the inverse transducer of M is the transducer $M' = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^9, \delta', \lambda' \rangle$ with memory of order $(1,2)$ defined by the infinite system

$$x_{t+2} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} y_t + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

Recall that the input symbols in this transducer are in $\mathcal{Y}$ and the output symbols in $\mathcal{X}$.

4.3 Criterium of Invertibility and Inverses of QLFTs with Memory

As said before, Renji Tao defined quasi-linear finite transducer in a way that the known results about invertibility of linear finite transducers could be extended [*Section* 3.4.2]. Let $\tau \in \mathbb{N}$ and $r \in \mathbb{N}_0$ such that $r \leq \tau$. As proved in *Theorem* 3.4.15, a τ-quasi-linear finite transducer is invertible with delay r if and only if the linear part of the transducer is invertible with delay r. So, the procedure presented before can be applied to quasi-linear finite transducers.

In the next example, it will be shown the procedure to check injectivity in a quasi-linear finite transducer with memory. This procedure could be applied only to the linear part of the transducer but, to be able to construct its inverse, it will be applied to the entire transducer.

Example 4.3.1. *Let* $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{12}, \delta, \lambda \rangle$ *be a* 2*-quasi-linear finite transducer with input memory of order* 4 *presented in* Example *3.4.14 and defined by:*

$$\begin{aligned} y_t &= A_0 x_{t+4} + A_1 x_{t+3} + A_2 x_{t+2} + f(x_t, x_{t+1}) \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1} \cdot x_t, \text{ for } t \geq 0, \end{aligned}$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$, $s_0 =< x_0, x_1, x_2, x_3 > \in \mathbb{F}_2^{12}$ *is the initial state of the transducer and* $\cdot$ *stands for componentwise multiplication. Let* $(x_t)_{t\geq 4}$ *be an input sequence and consider* $(y_t)_{t\geq 0} = \lambda(s_0, (x_t)_{t\geq 4})$.

Notice that, the third column of A_0 *is null. Therefore,* y_0 *does not contain any information about the third component of* x_4*. Consequently,* x_4 *is not uniquely determined by* y_0 *and* s_0*, i.e., the transducer is not invertible with delay* 0*. In fact, one just have to notice that* A_0 *does not have full rank.*

Since the transducer is not invertible with delay 0*, one has to apply a* $G_{(4,0)}$ *transformation. Then, the first step is to apply to the system a sequence of elementary row operations to obtain an equivalent system in reduced form, that is, to apply a* R_a *transformation. This can be obtained by adding the first and the second rows to the third one:*

$$\begin{bmatrix}1&0&0\\0&1&0\\1&1&1\end{bmatrix} y_t = \begin{bmatrix}1&1&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+4} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} x_{t+3} + \begin{bmatrix}1&0&0\\0&1&0\\0&1&1\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&1\\0&1&1\\0&1&0\end{bmatrix} x_{t+1}\cdot x_t.$$

There is one equation that does not depend on x_{t+4} and therefore can be discarded for the purpose of obtaining x_4 from s_0 and y_0y_1. The next step is to apply a R_b transformation, i.e, to discard the equations that do not depend on x_{t+4} and to reorganize the others by putting together the equations that depend on x_{t+4} and do not depend on subsequent inputs. To do that, for any two consecutive matricial equations in the system, reallocate the rows by putting together the first two rows of the first equation and the last row of the second equation. The result of this procedure is the following system:

$$\begin{bmatrix}0&0&0\\0&0&0\\1&1&1\end{bmatrix} y_{t+1} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix} x_{t+4} + \begin{bmatrix}1&0&0\\0&1&0\\0&1&1\end{bmatrix} x_{t+3} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+2} +$$
$$+ \begin{bmatrix}0&0&0\\0&0&0\\0&1&0\end{bmatrix} x_{t+2}\cdot x_{t+1} + \begin{bmatrix}1&0&1\\0&1&1\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

The coefficient matrix of x_{t+4} is invertible (has full rank), therefore, using this new system, x_4 is uniquely determined by s_0, y_0 and y_1, that is, the transducer is invertible with delay 1.

To construct an inverse transducer of a quasi-linear finite transducer, as in the linear case, one only has to multiply, on the left, the equation of the resulting system with full rank by the inverse of the coefficient matrix of x_h.

Example 4.3.2. *Let M be the quasi-linear finite transducer presented in the previous example. Consider the system which has full rank that defines the transducer. Since*

$$\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},$$

one can obtain an inverse transducer by multiplying the system with full rank by $L = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$*:*

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} +$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+2} \cdot x_{t+1} + \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} \cdot x_t, \text{ for } t \geq 0.$$

Then, the inverse transducer of M *is the transducer* $M' = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{15}, \delta', \lambda' \rangle$ *with memory of order* $(1, 4)$ *defined by the infinite system:*

$$x_{t+4} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_t + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} +$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+2} \cdot x_{t+1} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} \cdot x_t, \text{ for } t \geq 0.$$

As Renji Tao defined τ-quasi-linear finite transducers with memory of order (h, k), where $\tau \in \mathbb{N}$ and $h, k \in \mathbb{N}_0$, the criterium of invertibility of linear finite transducer with memory can be applied to them, i.e, a τ-QLFT with memory is invertible with delay $r \in \mathbb{N}_0$, $r \leq \tau$, if and only if, after applying r $G_{(h,k)}$ transformations, the system that defines the transducer has full rank. Quasi-linear finite transducers were defined this way, as far as we deduce, because, after a maximum of τ $G_{(h,k)}$ transformations, the input symbol x_h only appears in the linear part of the transducer.

For quasi-linear finite transducers defined in a general way, as in *Section* 3.4.2, we will have a different criterium. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a quasi-linear finite transducer with memory of order (h, k), $h, k \in \mathbb{N}_0$. To begin, M can be invertible with delay $\tau \in \mathbb{N}_0$, for $\tau = 0, 1, \ldots, h \dim(\mathcal{X})$, as linear finite transducers with memory, since the procedure to check invertibility is the same. However, to be able to recover x_h after τ $G_{(h,k)}$ transformations, in addition to the coefficient matrix of x_h having to be invertible, x_h can only appear in the linear

part. This new requirement is easy to understand. Suppose that, in the infinite system that defines the transducer M after τ $G_{(h,k)}$ transformations, one has x_h in the non-linear part of the transducer, for example, $x_h \cdot x_{h-1}$. The operation $\cdot$ is not reversible, since knowing x_{h-1} does not allow us to recover x_h. Therefore, x_h cannot appear in the non-linear part of the transducer.

Example 4.3.3. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \rangle$ be a quasi-linear finite transducer with input memory of order 2 defined by:*

$$\begin{aligned} y_t &= A_0 x_{t+2} + A_1 x_{t+1} + A_2 x_t + f(x_t, x_{t+1}) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} x_t + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} \cdot x_t, \text{ for } t \geq 0, \end{aligned}$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$ and $s_0 =< x_0, x_1 > \in \mathbb{F}_2^6$ is the initial state of the transducer. Let $(x_t)_{t\geq 2}$ be an input sequence and consider $(y_t)_{t\geq 0} = \lambda(s_0, (x_t)_{t\geq 2})$.

Notice that, the third column of A_0 is null. Therefore, y_0 does not contain any information about the third component of x_2. Consequently, the transducer is not invertible with delay 0.

Since the transducer is not invertible with delay 0, one has to apply a $G_{(2,0)}$ transformation. The system is in reduced form, so it is not necessary to apply a R_a transformation. There is one equation that does not depend on x_{t+2}, therefore one has to apply a R_b transformation to the infinite system:

$$\begin{cases} \quad \vdots & \\ y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} x_t + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} \cdot x_t & \\ y_{t+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} \cdot x_{t+1} & \\ \quad \vdots & \end{cases}.$$

To apply a R_b transformation, for any two consecutive matricial equations in the system, reallocate the rows by putting together the first two rows of the first equation and the last row of the second equation. The result of this procedure is the following system:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} y_{t+1} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}0&0&1\\1&0&1\\0&1&1\end{bmatrix} x_{t+1} +$$
$$+ \begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} x_t + \begin{bmatrix}1&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

The matrix A_0 does not have full rank therefore the transducer is not invertible with delay 1. It is necessary to apply another $G_{(2,0)}$ transformation. As the system is in reduced form, one only has to apply a R_b transformation. This way, one gets:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} y_{t+2} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&1&0\\0&1&1\end{bmatrix} x_{t+2} + \begin{bmatrix}0&0&1\\1&0&1\\0&0&0\end{bmatrix} x_{t+1} +$$
$$+ \begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} x_t + \begin{bmatrix}1&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

Now the system has full rank, therefore the transducer M is invertible with delay 2.

Example 4.3.4. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^9, \delta, \lambda \rangle$ be a quasi-linear finite transducer with input memory of order 3 defined by:*

$$y_t = A_0x_{t+3} + A_1x_{t+2} + A_2x_{t+1} + A_3x_t + f(x_t, x_{t+1}, x_{t+2})$$
$$= \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+3} + \begin{bmatrix}0&1&1\\0&0&1\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&1\\1&1&1\\0&0&0\end{bmatrix} x_{t+1} + \begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix} x_t + \begin{bmatrix}0&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t,$$

for $t \geq 0$, where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$ and $s_0 =< x_0, x_1, x_2 > \in \mathbb{F}_2^3$ is the initial state of the transducer. Let $(x_t)_{t\geq 3}$ be an input sequence and consider $(y_t)_{t\geq 0} = \lambda(s_0, (x_t)_{t\geq 3})$.

Since A_0 does not have full rank, the transducer is not invertible with delay 0. Therefore, one has to apply a $G_{(3,0)}$ transformation. The system is in reduced form, so it is only necessary to apply a R_b transformation. The resulting system is:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} y_{t+1} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+3} + \begin{bmatrix}0&1&1\\0&0&1\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&1\\1&1&1\\0&1&0\end{bmatrix} x_{t+1} +$$
$$+ \begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix} x_t + \begin{bmatrix}0&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

Notice that, adding the knowledge of y_{t+1} does not give any information about the third component of x_{t+3}, since the rank of the system is the same as before. The same will happen with y_{t+2}. After 3 $G_{(3,0)}$ transformations is obtained the following system:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} y_{t+3} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&1&0\\0&1&0\end{bmatrix} x_{t+3} + \begin{bmatrix}0&1&1\\0&0&1\\0&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&1\\1&1&1\\0&0&0\end{bmatrix} x_{t+1} +$$
$$+ \begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix} x_t + \begin{bmatrix}0&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

Lastly, one has to apply one more $G_{(3,0)}$ transformation. The resulting system of applying a R_a transformation and a R_b transformation is given by:

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} y_{t+4} + \begin{bmatrix}0&0&0\\0&0&0\\0&1&0\end{bmatrix} y_{t+1} + \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y_t = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} x_{t+3} + \begin{bmatrix}0&1&1\\0&0&1\\1&1&1\end{bmatrix} x_{t+2} +$$
$$+ \begin{bmatrix}1&0&1\\1&1&1\\1&0&1\end{bmatrix} x_{t+1} + \begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix} x_t + \begin{bmatrix}0&0&0\\0&0&0\\1&1&0\end{bmatrix} x_{t+2}\cdot x_{t+1} + \begin{bmatrix}0&1&1\\1&1&0\\0&0&0\end{bmatrix} x_{t+1}\cdot x_t, \text{ for } t \geq 0.$$

The resulting system has full rank, therefore the transducer M is invertible with delay 4. Notice that the delay is greater than the order of the memory.

Chapter 5

The Bao-Igarashi Attack to FAPCK

5.1 Composition of Finite Transducers

The composition of finite transducers is an essential operation on the cryptographic systems being discussed. The security of these cryptosystems is, in a sense, based on the problem of factoring non-linear finite transducers. Renji Tao presented, in his book [Tao09], two diferent compositions of finite transducers. In this section, it will be presented these compositions, that we call *usual composition* and *special composition*.

Definition 5.1.1. *Let* $M_i = \langle \mathcal{X}_i, \mathcal{Y}_i, S_i, \delta_i, \lambda_i \rangle$, $i = 1, 2$, *be two finite transducers with* $Y_1 = X_2$. *The* usual composition *of* M_1 *and* M_2, *denoted by* $M_2 \circ M_1$, *is the transducer*

$$M_2 \circ M_1 = \langle \mathcal{X}_1, \mathcal{Y}_2, S_1 \times S_2, \delta, \lambda \rangle$$

where, for $x \in \mathcal{X}_1$, $s_1 \in S_1$, *and* $s_2 \in S_2$,

$$\delta((s_1, s_2), x) = (\delta(s_1, x), \delta(s_2, \lambda(s_1, x)))$$
$$\lambda((s_1, s_2), x) = \lambda_2(s_2, \lambda_1(s_1, x)).$$

Basically, in the *usual composition*, the output of the first transducer M_1 is the input of the second transducer M_2.

To introduce the *special composition*, recall the *replacement map* defined in *Section* 3.3: let $\mathcal{X}$

be a non-empty set and $j \in \mathbb{N}$, the *replacement map* is defined by

$$\begin{aligned} \sigma : \ \mathcal{X}^j \times \mathcal{X} \ &\longrightarrow \mathcal{X}^j \\ ((x_1, x_2, \ldots, x_j), x) &\longmapsto (x_2, \ldots, x_j, x). \end{aligned}$$

Also, given a set $\mathcal{X}$, $n \in \mathbb{N}$, and $i, j \in \mathbb{N}$ such that $i+j \leq n+1$, let us define the (i,j)-*projection map*:

$$\begin{aligned} \pi_{i,j} : \ \mathcal{X}^n \ &\longrightarrow \mathcal{X}^j \\ (x_1, x_2, \ldots, x_n) &\longmapsto (x_i, x_{i+1}, \ldots, x_{i+j-1}). \end{aligned}$$

For any map $h : \mathcal{X}^{n+1} \longrightarrow \mathcal{Y}$ ($n \in \mathbb{N}_0$), and for any $m \in \mathbb{N}$, we will denote by h^{+m} the map $h^{+m} : \mathcal{X}^{n+m} \longrightarrow \mathcal{Y}^m$ given by $h^{+m}(x) = (h \circ \pi_{1,n+1}(x), h \circ \pi_{2,n+1}(x), \ldots, h \circ \pi_{m,n+1}(x))$, for any $x \in \mathcal{X}^{n+m}$.

Definition 5.1.2. *Let M_f and M_g be two finite transducers with memory induced by the functions*

$$f : \mathcal{X}^{h_f} \times \mathcal{X} \longrightarrow \mathcal{Y} \quad \textit{and} \quad g : \mathcal{Y}^{h_g} \times \mathcal{Z}^k \times \mathcal{Y} \longrightarrow \mathcal{Z},$$

i.e., M_f is a transducer with input memory of order h_f and M_g is a transducer with memory of order (h_g, k). The special composition *of M_f and M_g, denoted by $M_g \bullet M_f$, is the transducer with memory of order $(h_f + h_g, k)$*

$$M_g \bullet M_f = \left\langle \mathcal{X}, \mathcal{Z}, \mathcal{X}^{h_f+h_g} \times \mathcal{Z}^k, \delta, \lambda \right\rangle,$$

where δ and λ are given, for $x \in \mathcal{X}^{h_f+h_g}$, $z \in \mathcal{Z}^k$, and $a \in \mathcal{X}$, by

$$\begin{aligned} \delta((x,z),a) &= (\sigma(x,a), \sigma(z, \varphi(x,z,a))), \\ \lambda((x,z),a) &= \varphi(x,z,a), \end{aligned}$$

where $\varphi(x,z,a) = g(f \circ \pi_{1,h_f+1}(x), f \circ \pi_{2,h_f+1}(x), \ldots, f \circ \pi_{h_g,h_f+1}(x), z, f \circ \sigma(\pi_{h_g,h_f+1}(x), a))$ $= g(f^{+h_g}(x), z, f \circ \sigma(\pi_{h_g,h_f+1}(x), a)) \in \mathcal{Z}$.

One can now ask what is the relation between $M_g \circ M_f$ and $M_g \bullet M_f$. Renji Tao has shown the following result [Tao09, Theorem 1.2.1].

Proposition 5.1.3. *Let $M_f = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{h_f}, \delta_f, \lambda_f \rangle$ and $M_g = \langle \mathcal{Y}, \mathcal{Z}, \mathcal{Y}^{h_g} \times \mathcal{Z}^k, \delta_g, \lambda_g \rangle$ be as above. Then, for every state in $M_g \bullet M_f$ there is an equivalent state in $M_g \circ M_f$.*

Proof. Let $s = (x, z) \in \mathcal{X}^{h_f+h_g} \times \mathcal{Z}^k$ be a state of $M_g \bullet M_f$, and define

$$s_f = \pi_{h_g+1,h_f}(x), \quad s_g = (f^{+h_g}(x), z).$$

We will show that the state $(s_f, s_g) \in \mathcal{X}^{h_f} \times \mathcal{Y}^{h_g} \times \mathcal{Z}^k$ of $M_g \circ M_f$ is equivalent to the state s.

Note that $(s_f, a) = \sigma(\pi_{h_g,h_f+1}(x), a)$, and therefore

$$\begin{aligned}\lambda_\circ((s_f, s_g), a) &= \lambda_g(s_g, \lambda_f(s_f, a)) = \lambda_g(s_g, f(s_f, a)) = \lambda_g(s_g, f \circ \sigma(\pi_{h_g,h_f+1}(x), a)) \\ &= \lambda_g(f^{+h_g}(x), z, f \circ \sigma(\pi_{h_g,h_f+1}(x), a)) \\ &= g(f^{+h_g}(x), z, f \circ \sigma(\pi_{h_g,h_f+1}(x), a)) \\ &= \lambda_\bullet((x, z), a) = \lambda_\bullet(s, a).\end{aligned}$$

Also,

$$\begin{aligned}\delta_\circ((s_f, s_g), a) &= (\delta_f(s_f, a), \delta_g(s_g, \lambda_f(s_f, a))) \\ &= (\delta_f(s_f, a), \delta_g((f^{+h_g}(x), z), f(s_f, a))) \\ &= (\sigma(s_f, a), \sigma(f^{+h_g}(x), f(s_f, a)), \sigma(z, g(f^{+h_g}(x), z, f(s_f, a)))),\end{aligned}$$

while

$$\begin{aligned}\tilde{s} &= \lambda_\bullet(s, a) = \lambda_\bullet((x, z), a) \\ &= (\sigma(x, a), \sigma(z, g(f^{+h_g}(x), z, f \circ \sigma(\pi_{h_g,h_f+1}(x), a)))) \\ &= (\sigma(x, a), \sigma(z, g(f^{+h_g}(x), z, f(s_f, a)))) \\ &=: (\tilde{x}, \tilde{z}).\end{aligned}$$

Now, note that

$$\begin{aligned}\tilde{s}_f &= \pi_{h_g+1,h_f}(\tilde{x}) = \pi_{h_g+1,h_f}(\sigma(x, a)) = \sigma(\pi_{h_g+1,h_f}(x), a) = \sigma(s_f, a), \\ \tilde{s}_g &= (f^{+h_g}(\tilde{x}), \tilde{z}) = (f^{+h_g}(\sigma(x, a)), \sigma(z, g(f^{+h_g}(x), z, f(s_f, a)))).\end{aligned}$$

Hence, in order to show that $\delta_\circ((s_f, s_g), a) = (\tilde{s}_f, \tilde{s}_g)$, one only needs to check that $f^{+h_g}(\sigma(x, a)) = \sigma(f^{+h_g}(x), f(s_f, a))$:

$$\begin{aligned}\sigma(f^{+h_g}(x), f(s_f, a)) &= \sigma(f^{+h_g}(x), f(\pi_{h_g+1,h_f}(x), a)) \\ &= \sigma(f \circ \pi_{1,h_f+1}(x), f \circ \pi_{2,h_f+1}(x), \ldots, f \circ \pi_{h_g,h_f+1}(x), f(\pi_{h_g+1,h_f}(x), a)) \\ &= (f \circ \pi_{2,h_f+1}(x), \ldots, f \circ \pi_{h_g,h_f+1}(x), f(\pi_{h_g+1,h_f}(x), a)) = f^{+h_g}(\pi_{2,h_f}(x), a) \\ &= f^{+h_g}(\sigma(x, a)).\end{aligned}$$

□

Notice that, given an input sequence $\alpha \in \mathcal{X}^{h_f+h_g}$ and two equivalent states, $s \in \mathcal{X}^{h_f+h_g} \times \mathcal{Z}^k$ of $M_g \bullet M_f$ and $(s_f, s_g) \in \mathcal{X}^{h_f} \times \mathcal{Y}^{h_g} \times \mathcal{Z}^k$ of $M_g \circ M_f$, as defined above, both compound transducers produce the same output. Since for every state in $M_g \bullet M_f$ there is an equivalent state in $M_g \circ M_f$, one can use the usual composition instead of the special composition, if one chooses an initial state of $M_g \circ M_f$ that is equivalent to a state in $M_g \bullet M_f$.

In the FAPKC, the public key transducers are obtained using the special composition. Moreover, since every state has an equivalent state in the transducer obtained with the usual composition, in the FAPKC it is never used an initial state that is not equivalent to one in the public key transducer. Therefore, for now on, it will be used indistinguishably the usual and the special composition when refering to the public key transducer.

Let $M_f = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{h_f}, \delta_f, \lambda_f \rangle$ and $M_g = \langle \mathcal{Y}, \mathcal{Z}, \mathcal{Y}^{h_g} \times \mathcal{Z}^k, \delta_g, \lambda_g \rangle$ be two finite transducers. To compute the equation that defines the compound transducer M, one can use any of the compositions presented, since their output is the same. If one knows M_f and M_g, the compound transducer will be seen as $M_g \circ M_f$, because the *usual composition* is more natural. However, when one only knows M, it has to be seen as $M_g \bullet M_f$, because the symbols in $\mathcal{Y}$ have no meaning. In the case of knowing M_f and M_g, if one only knows the initial state $s \in \mathcal{X}^{h_f+h_g} \times \mathcal{Z}^k$ of M, then one has to find the equivalent state $(s_f, s_g) \in \mathcal{X}^{h_f} \times \mathcal{Y}^{h_g} \times \mathcal{Z}^k$ to be able to use the *usual composition*.

Example 5.1.4. *Let $M_f = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta_f, \lambda_f \rangle$ be a linear finite transducer with input memory of order 2 and $M_g = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{12}, \delta_g, \lambda_g \rangle$ be a linear finite transducer with memory of order $(1, 3)$. M_f and M_g are defined as follows:*

$$M_f : y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0.$$

$$M_g : z_{t+3} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} y_t + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} z_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} z_t, \text{ for } t \geq 0.$$

To compute the compound transducer $M = M_g \circ M_f$, one only has to substitute the $(y_t)_{t\geq 0}$ symbols in the equation of M_g by the relations of $(x_t)_{t\geq 0}$, obtained with the equation of M_f.

Since

$$\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix} y_{t+1} = \begin{bmatrix}1&1&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+3} + \begin{bmatrix}1&1&1\\0&1&1\\0&0&1\end{bmatrix} x_{t+2} + \begin{bmatrix}1&0&1\\0&0&0\\0&1&0\end{bmatrix} x_{t+1}, \text{ for } t \geq 0. \text{ and}$$

$$\begin{bmatrix}1&1&0\\0&0&0\\1&0&0\end{bmatrix} y_t = \begin{bmatrix}1&1&0\\0&0&0\\1&0&0\end{bmatrix} x_{t+2} + \begin{bmatrix}1&1&1\\0&0&0\\1&0&0\end{bmatrix} x_{t+1} + \begin{bmatrix}1&0&1\\0&0&0\\1&0&1\end{bmatrix} x_t, \text{ for } t \geq 0,$$

the compound transducer M *has memory of order* $(2+1, 3) = (3, 3)$ *and is given by:*

$$M : z_{t+3} = \begin{bmatrix}1&1&0\\0&1&0\\0&0&0\end{bmatrix} x_{t+3} + \begin{bmatrix}0&0&1\\0&1&1\\1&0&1\end{bmatrix} x_{t+2} + \begin{bmatrix}0&1&0\\0&0&0\\1&1&0\end{bmatrix} x_{t+1} + \begin{bmatrix}1&0&1\\0&0&0\\1&0&1\end{bmatrix} x_t +$$
$$+ \begin{bmatrix}1&0&1\\0&0&1\\0&0&0\end{bmatrix} z_{t+2} + \begin{bmatrix}1&0&0\\0&1&0\\0&1&0\end{bmatrix} z_t, \text{ for } t \geq 0.$$

Theorem 5.1.5. *Let* $M_0 = \langle \mathcal{X}, \mathcal{Y}, S_0, \delta_0, \lambda_0 \rangle$ *and* $M_1 = \langle \mathcal{Y}, \mathcal{Z}, S_1, \delta_1, \lambda_1 \rangle$ *be two finite transducers injective with delay* $\tau_0 \in \mathbb{N}_0$ *and* $\tau_1 \in \mathbb{N}_0$*, respectively. The compound transducer* $M = M_1 \circ M_0 = \langle \mathcal{X}, \mathcal{Z}, S, \delta, \lambda \rangle$ *is injective with delay* $\tau_0 + \tau_1$*.*

Proof. The transducer M is injective with delay $\tau_0 + \tau_1$ if

$$\forall s \in S,\ \forall x, x' \in \mathcal{X},\ \forall \mu, \mu' \in \mathcal{X}^{\tau_0+\tau_1}, \quad \lambda(s, x\mu) = \lambda(s, x'\mu') \implies x = x'.$$

Since the transducers M_0 and M_1 are injective with delay τ_0 and τ_1, respectively, one has:

$$\forall s_0 \in S_0,\ \forall x, x' \in \mathcal{X},\ \forall \alpha, \alpha' \in \mathcal{X}^{\tau_0}, \quad \lambda_0(s_0, x\alpha) = \lambda_0(s_0, x'\alpha') \implies x = x',$$
$$\forall s_1 \in S_1,\ \forall y, y' \in \mathcal{Y},\ \forall \beta, \beta' \in \mathcal{Y}^{\tau_1}, \quad \lambda_1(s_1, y\beta) = \lambda_1(s_1, y'\beta') \implies y = y'.$$

Let $x, x' \in \mathcal{X}$, $\alpha_{\tau_0}, \alpha'_{\tau_0} \in \mathcal{X}^{\tau_0}$, $\alpha_{\tau_1}, \alpha'_{\tau_1} \in \mathcal{X}^{\tau_1}$, $\mu = \alpha_{\tau_0}\alpha_{\tau_1} \in \mathcal{X}^{\tau_0+\tau_1}$ and $\mu' = \alpha'_{\tau_0}\alpha'_{\tau_1} \in \mathcal{X}^{\tau_0+\tau_1}$. Let $s_0 \in S_0$, $s_1 \in S_1$ and $s = (s_0, s_1) \in S$. Recall the definition of the output function in the usual composition presented in *Definition* 5.1.1: $\lambda((s_0, s_1), x) = \lambda_1(s_1, \lambda_0(s_0, x))$. Then, the following statements are equivalent:

$$\lambda(s, x\mu) = \lambda(s, x'\mu')$$
$$\lambda_1(s_1, \lambda_0(s_0, x\mu)) = \lambda_1(s_1, \lambda_0(s_0, x'\mu'))$$
$$\lambda_1(s_1, \lambda_0(s_0, x\alpha_{\tau_0}\alpha_{\tau_1})) = \lambda_1(s_1, \lambda_0(s_0, x'\alpha'_{\tau_0}\alpha'_{\tau_1})).$$

Let $\lambda_0(s_0, x\alpha_{\tau_0}\alpha_{\tau_1}) = y\beta_{\tau_0}\beta_{\tau_1}$ and $\lambda_0(s_0, x'\alpha'_{\tau_0}\alpha'_{\tau_1}) = y'\beta'_{\tau_0}\beta'_{\tau_1}$, for some $y, y' \in \mathcal{Y}$, $\beta_{\tau_0}, \beta_{\tau_0} \in \mathcal{Y}^{\tau_0}$ and $\beta_{\tau_1}, \beta_{\tau_1} \in \mathcal{Y}^{\tau_1}$. M_1 is injective with delay τ_1, then $y = y'$ and $\beta_{\tau_0} = \beta'_{\tau_0}$. It follows that

$$y\beta_{\tau_0} = y'\beta'_{\tau_0} \iff \lambda_0(s_0, x\alpha_{\tau_0}) = \lambda_0(s_0, x'\alpha'_{\tau_0}).$$

Since M_0 is injective with delay τ_0, one has $x = x'$. One can conclude that M is injective with delay $\tau_0 + \tau_1$. □

Notice that, in the previous proof, it was used the formula of the output function in the usual composition. However, it was proved in *Proposition* 5.1.3 that the output of the compound transducer is the same in the usual and in the special composition, therefore, the result is also valid for the special composition.

Example 5.1.6. *The transducers M_0 and M_1 presented in* Example *5.1.4 are invertible with delay 1 and 0, respectively. The transducer $M = M_1 \circ M_0$ is invertible with delay 1.*

5.2 General Discription of FAPKCs

The first FAPKC system was proposed in 1985 by Tao and Chen in a paper (in Chinese) and was named FAPKC0. An English description of it was presented in a later work of the same authors [TC86]. In this system, the private key is composed by the inverses of two injective transducers with memory, where one is a linear finite transducer τ-injective ($\tau > 15$), and the other is a quasi-linear finite transducer with delay 0. This inverses can be easily computed, as seen in the last chapter. The public key is the result of the composition of the original pair, thus obtaining a non-linear finite transducer. In 1986, Tao and Chen published two variants of the cryptosystem FAPKC0, named FAPKC1 and FAPKC2 [TC86]. In FAPKC1, the two finite transducers whose inverses compose the private key have the same characteristics that in FAPKC0. But, in FAPKC2, the quasi-linear transducer is invertible with some delay different than zero. Later, two new cryptographic schemes appeared: FAPKC3 and FAPKC4, presented by Tao et al. [TCC97] and by Tao and Chen [TC97], respectively. Meanwhile, some other schemes of Public Key Cryptography based on finite transducers were developed (the system FAPKC93 was presented in a PhD thesis written in Chinese, and a variant of FAPKC2 was put forward by Bao and Igarashi [BI95]). All of these systems are similar in structure, their main difference being the choice of the transducers for the private key.

The crucial point of these cryptosystems' security is that it is easy to obtain an inverse of the compound transducer from the inverses of its factors, while it is believed to be hard to find that inverse without knowing those factors. On the other hand, the factorization of a transducer seems to be hard by itself.

It is known that, if one of the finite transducers that originate the private key is linear and the other is invertible with delay 0, the cryptosystem is insecure [BI95] (which is the case of FAPKC0, FAPKC1, and FAPKC93). However, if the two finite transducers are quasi-linear or one of them is linear and the other's delay is greater than 0, the FAPKCs seem secure whenever a so-called linear $R_a\ R_b$ check process is included in the key-generator [Tao95]. FAPKC2, FAPKC3, and FAPKC4 are examples of such FAPKCs.

In the next section, it will be presented the Bao-Igarashi attack to FAPKC. In order to understand the attack, first, let us properly introduce these cryptosystems.

5.2.1 Cryptosystems FAPKC

In all FAPKCs, the private key is composed by the inverses of two or more finite transducers and their initial states. The public key is given by the composition of all finite transducers whose inverses are in the private key, as well as its initial state. Usually, the input and the output are 8-dimensional vectors over $\mathbb{F}_2$, since all characters can be represented by a byte. The main difference between these cryptosystems is the type of the transducers that originate the private key. The FAPKC as presented in this section is the base of all FAPKCs. Although it has been proved as unsafe (the attack will be presented in the next section), this scheme is one of the few for which it is possible to construct examples. For the versions pointed out as safer, there are no examples available and the method presented by the author is not clear.

The procedure of generating key pairs is the following:

1. Choose a quasi-linear finite transducer with memory, M_0, invertible with delay 0. Compute an inverse transducer of M_0, M_0^{-1};

2. Choose a linear finite transducer with memory, M_1, invertible with delay τ (typically $\tau > 15$). Compute an inverse transducer of M_1, denoted by M_1^{-1};

3. Compute the compound transducer M from M_0 and M_1, and choose initial state s_M;

4. The private key is the pair (M_0^{-1}, M_1^{-1}). The public key is the compound transducer and its initial state, i.e., (M, s_M).

In FAPKC, a plaintext is encrypted using the public key transducer M, a non-linear finite transducer with delay τ. The plaintext is the input sequence of M, and the output is the ciphertext. Notice that, since the transducer is invertible with delay τ, one needs to append the plaintext with τ arbitrarily chosen symbols for the recipient to be able to recover the full message.

To decrypt the ciphertext, one only has to use the inverse transducers M_0^{-1} and M_1^{-1} in the private key with the initial states obtained from the inverse states of the ones that compound the equivalent state of s_M for the *usual composition*. Other possible way to decrypt the ciphertext is by computing the inverse transducer of M from M_0^{-1} and M_1^{-1}, $M^{-1} = M_0^{-1} \circ M_1^{-1}$, and its initial state (the inverse state of s_M).

The principle of encryption and decryption of FAPKC is as shown in the next figures, where the plaintext is a sequence of length $m+1$. M_0 is a quasi-linear transducer with input memory of order $h_0 \in \mathbb{N}_0$ and delay 0, M_1 is a linear transducer with memory of order (h_1, k), $h_1, k \in \mathbb{N}_0$, and delay $\tau \in \mathbb{N}$, and M_0' and M_1' are the respective inverses. Futhermore, M is the compound transducer of M_0 and M_1, and M' its inverse.

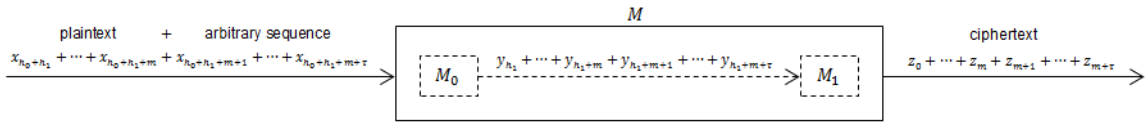


Figure 5.1: Principle of Encryption of FAPKC

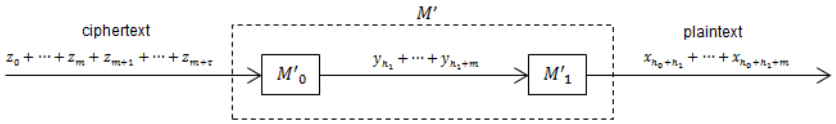


Figure 5.2: Principle of Decryption of FAPKC

Next, it will be presented two simple examples of the encrypt and decrypt procedures on FAPKC. Let $M_0 = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta_0, \lambda_0 \rangle$ and $M_1 = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta_1, \lambda_1 \rangle$ be the transducers defined as follows:

$$M_0 : y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

$$M_1 : z_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_{t+2} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_t, \text{ for } t \geq 0.$$

M_0 is a quasi-linear finite transducer with input memory of order 2 and invertible with delay 0, and M_1 is a linear transducer with input memory 2 and invertible with delay 2. To compute the compound transducer $M = M_1 \circ M_0$, one only has to substitute the $(y_t)_{t\geq 0}$ symbols in the equation of M_1 by the equivalent relations of $(x_t)_{t\geq 0}$, obtained with the equation of M_0:

$$M : z_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

The compound transducer is a non-linear finite transducer with input memory of order 4 and invertible with delay 2. Then, the public key is composed by the transducer M and some initial state s, for example, $s =< x_0, x_1, x_2, x_3 >=< \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} >$.

Example 5.2.1. *If one wants to encryt a message to the owner of the public key presented before, one just needs to compute the output using the transducer M and the initial state given.*

For example, to encrypt $\alpha = x_4x_5 = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\end{bmatrix}$*, one has:*

$$z_0 = \lambda(< x_0, x_1, x_2, x_3 >, x_4) = \lambda\left(< \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} >, \begin{bmatrix}1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}1\\0\\0\end{bmatrix}.$$

$$z_1 = \lambda(< x_1, x_2, x_3, x_4 >, x_5) = \lambda\left(< \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix} >, \begin{bmatrix}0\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\1\\0\end{bmatrix}.$$

But, since the transducer has delay 2, one needs to append the plaintext with 2 arbitrarily chosen symbols, say $\beta = x_6x_7 = \begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}$*, and calculate the respective output:*

$$z_2 = \lambda(< x_2, x_3, x_4, x_5 >, x_6) = \lambda\left(< \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix} >, \begin{bmatrix}0\\0\\1\end{bmatrix}\right) = \begin{bmatrix}1\\0\\1\end{bmatrix}.$$

$$z_3 = \lambda(< x_3, x_4, x_5, x_6 >, x_7) = \lambda\left(< \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} >, \begin{bmatrix}1\\1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix}.$$

Then, the ciphertext is $\lambda(s, \alpha\beta) = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}$.

For the decryption process, it is necessary to compute the inverses of the transducers M_0 and M_1, which will be the private key. To do that, one just needs to aplicate the procedure presented in *Chapter* 4. So, the inverse transducers are defined by:

$$M_0^{-1} : x_{t+2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_t + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

$$M_1^{-1} : y_{t+2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} z_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} z_t + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} y_{t+1}, + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_t, \text{ for } t \geq 0.$$

Notice that M_0^{-1} is a quasi-linear finite transducer with memory of order $(0, 2)$ and M_1^{-1} is a linear finite transducer with memory $(2, 2)$. The fact of M_1^{-1} being a transducer with input memory equals to the delay of M_1 is the reason why we have to append the plaintext with 2 arbitrarily chosen symbols.

Lastly, we need the initial states of these transducers. The initial state of M_1^{-1} is $< y_0, y_1, z_0, z_1 >=$ $< \lambda_0(< x_0, x_1 >, x_2), \lambda_0(< x_1, x_2 >, x_3), z_0, z_1 >=< \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} >$ and the initial state of M_0^{-1} is $< x_2, x_3 >$ (it is obvious if one notices that the first input is x_4).

Example 5.2.2. *To decrypt the ciphertext computed in the previous example, the owner of the public key just needs to use the inverses of the transducers in the private key and the respective initial states. In the first place, one has to use M_1^{-1} to obtain y_2 and y_3:*

$$y_2 = \lambda_1^{-1}(< y_0, y_1, z_0, z_1 >, z_2) = \lambda_1^{-1}\left(< \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} >, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

$$y_3 = \lambda_1^{-1}(< y_1, y_2, z_1, z_2 >, z_3) = \lambda_1^{-1}\left(< \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} >, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Therefore, using M_0^{-1}, one can recover the input $\alpha = x_4x_5$:

$$x_4 = \lambda_0^{-1}(< x_2, x_3 >, y_2) = \lambda_0^{-1}\left(< \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} >, \begin{bmatrix}0\\0\\1\end{bmatrix} \right) = \begin{bmatrix}1\\0\\0\end{bmatrix}.$$

$$x_5 = \lambda_0^{-1}(< x_3, x_4 >, y_3) = \lambda_0^{-1}\left(< \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix} >, \begin{bmatrix}0\\0\\0\end{bmatrix} \right) = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

5.3 The Bao-Igarashi Attack

Bao and Igarashi proved, in their paper [BI95], a previously unknow property of invertible finite transducers. By this property, they reduce the problem of separating M_0 and M_1 from the compound transducer $M = M_1 \circ M_0$ to a much easier problem: finding a special left common factor of two given matrix polinomials in module matrix polynomial rings. Roughly speaking, given a public key transducer, this property allow us to find two finite transducers M_0' and M_1' such that, the inverse transducer of $M' = M_1' \circ M_0'$, inverts M as well.

Let us start by proving a property of invertible finite transducers: only the $\tau + 1$ most recent inputs determine whether a transducer is invertible with delay $\tau \in \mathbb{N}_0$. It will be used transducers with input-only-memory, since the problem of checking injectivity of finite transducers with input and output memory can be reduced to the problem of checking injectivity of transducers with input-only-memory (*Theorem* 3.4.5). Let $\tau, h \in \mathbb{N}_0$. Let $M_f = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{\tau+h}, \delta_f, \lambda_f \rangle$ and $M_{f+g} = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{\tau+h}, \delta_{f+g}, \lambda_{f+g} \rangle$ be a pair of transducers with input memory of order $\tau + h$, defined as follows:

$$M_f : y_t = f(x_{t+\tau+h}, x_{t+\tau+h-1}, \ldots, x_t), \text{ for } t \geq 0,$$
$$M_{f+g} : y_t = f(x_{t+\tau+h}, x_{t+\tau+h-1}, \ldots, x_t) + g(x_{t+h-1}, \ldots, x_t), \text{ for } t \geq 0,$$

where $f : \mathcal{X}^{\tau+h+1} \longrightarrow \mathcal{Y}$ and $g : \mathcal{X}^h \longrightarrow \mathcal{Y}$. Notice that the $\tau + 1$ most recent inputs, $x_{t+\tau+h}, x_{t+\tau+h-1}, \ldots, x_{t+h}$, only appear in function f.

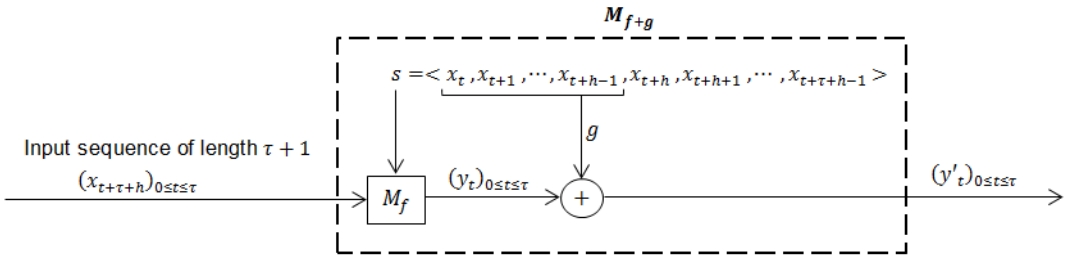


Figure 5.3: Relation between M_f and M_{f+g}

Theorem 5.3.1. *The finite transducer M_f is invertible with delay τ if and only if M_{f+g} is invertible with delay τ.*

Proof. A transducer $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{\tau+h}, \delta, \lambda \rangle$ is τ-injective if

$$\forall s \in \mathcal{X}^{\tau+h},\ \forall x, x' \in \mathcal{X},\ \forall \alpha, \alpha' \in \mathcal{X}^{\tau},\quad \lambda(s, x\alpha) = \lambda(s, x'\alpha') \implies x = x'.$$

Since, $\forall s =< x_0, \ldots, x_{h-1}, x_h, \ldots, x_{h+\tau-1} > \in \mathcal{X}^{h+\tau}$, $\forall x, x' \in \mathcal{X}$, one has:

$$\begin{aligned}
\lambda_{f+g}(s, x) &= \lambda_{f+g}(s, x') \\
f(x, x_{\tau+h-1}, \ldots, x_0) + g(x_{h-1}, \ldots, x_0) &= f(x', x_{\tau+h-1}, \ldots, x_0) + g(x_{h-1}, \ldots, x_0) \\
f(x, x_{\tau+h-1}, \ldots, x_0) &= f(x', x_{\tau+h-1}, \ldots, x_0) \\
\lambda_f(s, x) &= \lambda_f(s, x').
\end{aligned}$$

To prove that $\forall x, x' \in \mathcal{X}$, $\forall \alpha, \alpha' \in \mathcal{X}^{\tau}$ one has

$$\lambda_{f+g}(s, x\alpha) = \lambda_{f+g}(s, x'\alpha') \iff \lambda_f(s, x\alpha) = \lambda_f(s, x'\alpha'),$$

just notice that the input word has length $\tau + 1$ so, the inputs that appear in function g are all part of s, i.e, are constant. Therefore, they cancel each other in $\lambda_{f+g}(s, x\alpha) = \lambda_{f+g}(s, x'\alpha')$ leaving with $\lambda_f(s, x\alpha) = \lambda_f(s, x'\alpha')$.

Thus, M_f is invertible with delay τ if and only if M_{f+g} is invertible with delay τ. □

Next, we show that it is possible to construct an inverse transducer of M_{f+g} from an inverse transducer of M_f.

Let $M_f^{-1} = \left\langle \mathcal{Y}, \mathcal{X}, \mathcal{Y}^{\tau} \times \mathcal{X}^{\tau+h}, \delta_f^{-1}, \lambda_f^{-1} \right\rangle$ be an inverse transducer with delay τ of M_f. Let $s =< x_0, x_1, \ldots, x_{\tau+h-1} >$ be the initial state of M_f and $x\alpha \in \mathcal{X} \times \mathcal{X}^{\tau}$ be an input sequence. If $y = y_0 y_1 \ldots y_{\tau} \in \mathcal{Y}^{\tau+1}$ is the output of M_f, i.e, $y = \lambda_f(s, x\alpha)$, and $s^{-1} =$

$< y_0, y_1, \ldots, y_{\tau-1}, x_0, x_1, \ldots, x_{\tau+h-1} >$, then M_f^{-1} can recover x because $x = \lambda_f^{-1}(s^{-1}, y_\tau)$. The output $y' = y'_0 y'_1 \ldots y'_\tau \in \mathcal{Y}^{\tau+1}$ of M_{f+g}, for the same input sequence, is given by

$$y'_t = y_t + g(x_{t+h-1}, \ldots, x_t), \text{ for } 0 \leq t \leq \tau.$$

To invert M_{f+g}, given y', one just needs to compute $y_t = y'_t - g(x_{t+h-1}, \ldots, x_t)$, $0 \leq t \leq \tau$, and use M_f^{-1} to recover x.

The principle of constructing M_{f+g}^{-1} is illustrated in the next figure.

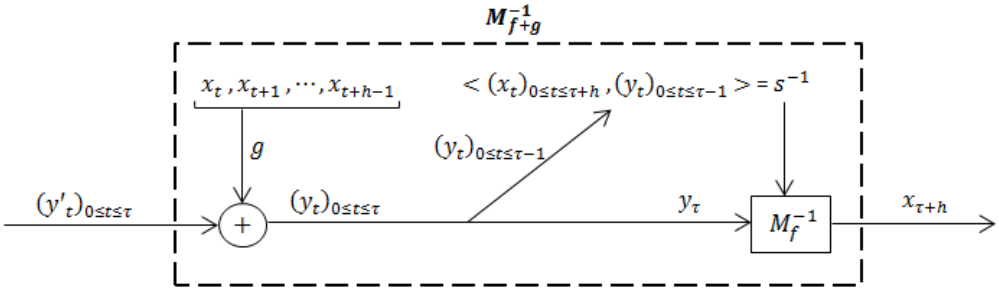


Figure 5.4: Principle of constructing M_{f+g}^{-1}

Notice that, if one has two transducers M_f and M_{f+g} in the previous conditions, then one can see any of them as the base transducer or the extended one. Let $M_{f'} = M_{f+g}$, i.e., $f' : \mathcal{X}^{\tau+h+1} \longrightarrow \mathcal{Y}$ such that $f' = f + g$. One can see M_f as $M_{f'+g'}$ where $g' : \mathcal{X}^h \longrightarrow \mathcal{Y}$ is such that $g' = -g$.

Example 5.3.2. *Let $M_f = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^9, \delta_f, \lambda_f \rangle$ be the transducer with input memory of order 3 defined by:*

$$\begin{aligned} y_t &= f(x_{t+3}, x_{t+2}, x_{t+1}, x_t) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0, \end{aligned}$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$, $(y_t)_{t\geq 0} \in \mathbb{F}_2^3$ and $f : \mathbb{F}_2^{12} \to \mathbb{F}_2^3$. M_f is invertible with delay 1 and its inverse transducer, $M_f^{-1} = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{12}, \delta_f^{-1}, \lambda_f^{-1} \right\rangle$, is given by:

$$x_{t+3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_t + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_t.$$

Let $M_{f+g} = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^9, \delta_{f+g}, \lambda_{f+g} \right\rangle$ *be the transducer with input memory of order* 3 *defined by:*

$$y'_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0,$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$ *and* $(y'_t)_{t\geq 0} \in \mathbb{F}_2^3$. *If one considers* $g : \mathbb{F}_2^6 \to \mathbb{F}_2^3$ *where*

$$g(x_t, x_{t+1}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t,$$

then $y'_t = f(x_{t+3}, x_{t+2}, x_{t+1}, x_t) + g(x_t, x_{t+1})$, *for* $t \geq 0$. *Let* $s =< x_0, x_1, x_2 >=< \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} >$ *be the inicial state of* M_{f+g} *(also the inicial state of* M_f*) and* $\alpha = x_3x_4 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ *be the input sequece. Then, the output of* M_{f+g} *is:*

$$y'_0y'_1 = \lambda_{f+g}(s, \alpha) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Since $y_t = y'_t - g(x_t, x_{t+1})$, *for* $t = 0, 1$, *one has:*

$$y_0 = y'_0 - g(x_0, x_1) = y'_0 - g\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

$$y_1 = y'_1 - g(x_1, x_2) = y'_1 - g\left(\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Let $s^{-1} =< y_0, x_0, x_1, x_2 >$ *be the initial state of* M_f^{-1} *and let* y_1 *be the input. Then, the output is:*

$$\lambda_f^{-1}(s^{-1}, y_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = x_3.$$

This way, one can recover the input symbol x_3 *of* M_{f+g} *using the inverse transducer of* M_f.

Now, we are in condition to present the Bao-Igarashi attack to FAPKC.

Let M_0 and M_1 be the transducers whose inverses are in the private key. Let $M_0 = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^{h_0}, \delta_0, \lambda_0 \rangle$ be a quasi-linear finite transducer with input memory h_0 and invertible with delay 0 defined by

$$M_0 : \ y_t = \sum_{j=0}^{h_0} B_j x_{t+h_0-j} + \sum_{j=1}^{h_0-1} \tilde{B}_j x_{t+h_0-j} \cdot x_{t+h_0-j-1}, \text{ for } t \geq 0,$$

where, for some $\ell \in \mathbb{N}$, $(B_j)_{0\leq j\leq h_0}$,$(\tilde{B}_j)_{1\leq j\leq h_0-1} \in \mathcal{M}_\ell(\mathbb{F}_2)$ and B_0 is an invertible matrix. The operation $\cdot$ is defined to be componentwise multiplication, but could be any other non-linear binary operation. Notice that, x_{t+h} only appears in the linear part of M_0. It is easy to see that the transducer M_0^{-1}, with memory of order $(0, h)$, given by:

$$M_0^{-1} : \ x_{t+h_0} = B_0^{-1} \left(y_t + \sum_{j=1}^{h_0} B_j x_{t+h_0-j} + \sum_{j=1}^{h_0-1} \tilde{B}_j x_{t+h_0-j} \cdot x_{t+h_0-j-1} \right), \text{ for } t \geq 0,$$

is an inverse transducer with delay 0 of M_0. For any initial state $s_0 =< x_0, x_1, \ldots, x_{h-1} > \in \mathcal{X}^h$ of M_0, its inverse state in M_0^{-1} is also $< x_0, x_1, \ldots, x_{h-1} >$.

Let $M_1 = \langle \mathcal{Y}, \mathcal{Z}, \mathcal{Y}^{h_1}, \delta_1, \lambda_1 \rangle$ be a linear transducer with input memory h_1 and invertible with delay $\tau \in \mathbb{N}$, defined by

$$M_1 : \ z_t = \sum_{i=0}^{h_1} A_i y_{t+h_1-i}, \text{ for } t \geq 0 \text{ and } A_i \in \mathcal{M}_\ell(\mathbb{F}_2).$$

The compound transducer $M = M_1 \circ M_0$ is obtained by substituting $(y_t)_{t\geq 0}$ in the definition of M_1 by the ones given by the equation of M_0 (as seen in the previous section).

$$M : \ z_t = \sum_{i=0}^{h_1} A_i \left(\sum_{j=0}^{h_0} B_j x_{t+h_0+h_1-j-i} + \sum_{j=1}^{h_0-1} \tilde{B}_j x_{t+h_0+h_1-j-i} \cdot x_{t+h_0+h_1-j-i-1} \right), \text{ for } t \geq 0.$$

This equation can be simplified as follows:

$$M : \ z_t = \sum_{k=0}^{h_0+h_1} C_k x_{t+h_0+h_1-k} + \sum_{k=1}^{h_0+h_1-1} \tilde{C}_k x_{t+h_0+h_1-k} \cdot x_{t+h_0+h_1-k-1}, \text{ for } t \geq 0,$$

where

$$C_k = \sum_{i+j=k} A_i B_j, \text{ for } k = 0, 1, \ldots, h_0 + h_1; \text{ and}$$

$$\tilde{C}_k = \sum_{i+j=k} A_i \tilde{B}_j, \text{ for } k = 1, 2, \ldots, h_0 + h_1 - 1.$$

In fact, the Bao-Igarashi Attack only works when the delay of the transducer M_1 is such that $\tau \leq h_0$ and $\tau \leq h_1 - 1$. Here, it will be presented a generalization of the attack. In order to do that, one has to extend the input memory of the transducers, with as many null matrices as necessary, in order that τ verifies the previous conditions. Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h, \delta, \lambda \rangle$ be a finite transducer with input memory h defined by

$$y_t = \sum_{i=0}^{h} A_i x_{t+h-i}, \text{ for } t \geq 0,$$

where $(x_t)_{t\geq 0} \in \mathcal{X}$, $(y_t)_{t\geq 0} \in \mathcal{Y}$ and $s =< x_0, x_1, \ldots, x_{h-1} >$ is the initial state. The extended transducer $M^* = \left\langle \mathcal{X}, \mathcal{Y}, 0^{\ell}_{\dim(\mathcal{X})\times 1} \times \mathcal{X}^h, \delta^*, \lambda^* \right\rangle$ with input memory of order $h + \ell$ is given by

$$y_t = \sum_{i=0}^{h+\ell} A'_i x_{t+h+\ell-i}, \text{ for } t \geq 0,$$

where $A'_i = A_i$ for $i = 0, 1, \ldots, h$ and $A'_i = 0$ for $i = h+1, h+2, \ldots, h+\ell$, i.e, the extended transducer is defined by

$$y_t = \sum_{i=0}^{h} A_i x_{t+h+\ell-i}, \text{ for } t \geq 0.$$

The initial state of this transducer is also appended with ℓ null vectors, so it is given by $< 0, \ldots, 0, x_0, x_1, \ldots, x_{h-1} > \in 0^{\ell}_{\dim(\mathcal{X})\times 1} \times \mathcal{X}^h$. Let $\mathbf{0} =< 0, \ldots, 0 > \in 0^{\ell}_{\dim(\mathcal{X})\times 1}$. The state transition function δ^* is defined as follows

$$\delta^*(< \mathbf{0}, s >, x) =< \mathbf{0}, \delta(s, x) >, \text{ where } x \in \mathcal{X} \text{ and } s \in \mathcal{X}^h.$$

It is obvious that the transducer M and the extended transducer M^* are equivalent.

Example 5.3.3. *Let $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^6, \delta, \lambda \right\rangle$ be a finite transducer with memory of order $(2, 0)$ defined by*

$$y_t = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_t, \text{ for } t \geq 0,$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$, $(y_t)_{t\geq 0} \in \mathbb{F}_2^3$ and $s_0 =< x_0, x_1, x_2 >$ is the initial state. The equivalent transducer with input memory of order 5 is given by

$$y_t = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+4} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_t,$$

where $(x_t)_{t\geq 0} \in \mathbb{F}_2^3$, $(y_t)_{t\geq 0} \in \mathbb{F}_2^3$ *and* $s_0 =< 0_{3\times 1}, 0_{3\times 1}, x_0, x_1, x_2 >$ *is the initial state of the transducer.*

Let $\mathcal{A}$, $\mathcal{B}$, $\tilde{\mathcal{B}}$, $\mathcal{C}$, $\tilde{\mathcal{C}}$, be the following sets of matrices in the equations that define the transducers M_0, M_1 and M, as presented before:

$$\mathcal{A} = \{A_i : 0 \leq i \leq h_1\}, \qquad \mathcal{C} = \{C_k : 0 \leq k \leq h_0 + h_1\},$$
$$\mathcal{B} = \{B_j : 0 \leq j \leq h_0\}, \qquad \tilde{\mathcal{C}} = \{\tilde{C}_k : 1 \leq k \leq h_0 + h_1 - 1\},$$
$$\tilde{\mathcal{B}} = \{\tilde{B}_j : 1 \leq j \leq h_0 - 1\}.$$

Suppose that, for the sets of matrices $\mathcal{C}$ and $\tilde{\mathcal{C}}$, one can find a new set of matrices $\mathcal{A}' = \{A'_i : 0 \leq i \leq \tau\}$, $\mathcal{B}' = \{B_j : 0 \leq j \leq \tau\}$ (where B'_0 is invertible) and $\tilde{\mathcal{B}}' = \{\tilde{B}'_j : 1 \leq j \leq \tau\}$ such that $C_k = \sum_{i+j=k} A'_i B'_j$, for $k = 0, 1, ..., \tau$, and $\tilde{C}_k = \sum_{i+j=k} A'_i \tilde{B}'_j$, for $k = 1, ..., \tau$.

It is easy to construct a quasi-linear finite transducer, M'_0, from the sets $\mathcal{B}'$ and $\tilde{\mathcal{B}}'$. This transducer is invertible with delay 0, since B'_0 is an invertible matrix. It is also easy to construct a linear transducer, M'_1, from $\mathcal{A}'$. Then, one can construct the transducer $M' = M'_1 \circ M'_0$.

$$M' : \; z_t = \sum_{k=0}^{2\tau} C'_k x_{t+h_0+h_1-k} + \sum_{k=1}^{2\tau} \tilde{C}'_k x_{t+h_0+h_1-k} \cdot x_{t+h_0+h_1-k-1}, \text{ for } t \geq 0,$$

where $C'_k = \sum_{i+j=k} A'_i B'_j$, for $k = 0, 1, \ldots, 2\tau$ and $\tilde{C}'_k = \sum_{i+j=k} A'_i \tilde{B}'_j$, for $k = 1, 2, \ldots, 2\tau$. Since $C_k = C'_k$ for $k = 0, 1, \ldots, \tau$ and $\tilde{C}_k = \tilde{C}'_k$ for $k = 1, 2, \ldots, \tau$, the transducers M and M' have the same coefficient matrices for the $\tau + 1$ most recent inputs. This means that, from *Theorem* 5.3.1 and since M is invertible with delay τ, M' is also invertible with delay τ. Thus, as illustrated in *Example* 5.3.2, one can contruct an inverse transducer with delay τ of M from $M'^{-1} = M'^{-1}_0 \circ M'^{-1}_1$.

The problem of finding such sets of matrices $\mathcal{A}'$, $\mathcal{B}'$ and $\tilde{\mathcal{B}}'$ can be solved as follows. Let B'_0 be the identity matrix and $B'_j = 0$ for $j = 1, 2, \ldots, \tau$. Then, since one wants $C_k = \sum_{i+j=k} A'_i B'_j$, one just needs to choose $A'_i = C_i$, for $i = 0, 1, ..., \tau$. One can find the set of matrices $\tilde{\mathcal{B}}'$ such that $\tilde{C}_k = \sum_{i+j=k} A'_i \tilde{B}'_j = \sum_{i+j=k} C_i \tilde{B}'_j$, for $k = 1, ..., \tau$, by solving the following system of linear equations

$$\begin{bmatrix} C_0 & 0 & \cdots & 0 \\ C_1 & C_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{\tau-1} & C_{\tau-2} & \cdots & C_0 \end{bmatrix} \begin{bmatrix} \tilde{B}'_1 \\ \tilde{B}'_2 \\ \vdots \\ \tilde{B}'_\tau \end{bmatrix} = \begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \vdots \\ \tilde{C}_\tau \end{bmatrix}.$$

This system definitely has solutions. Actually, one simple solution is obtained using the facts that $\tilde{C}_k = \sum_{i+j=k} A_i \tilde{B}_j$, for $k = 1, 2, \ldots, \tau$, and $C_k = \sum_{i+j=k} A_i B_j$, for $k = 0, 1, 2, \ldots, \tau$, as follows:

$$\begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \\ \vdots \\ \tilde{C}_\tau \end{bmatrix} = \begin{bmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_{\tau-1} & A_{\tau-2} & \cdots & A_0 \end{bmatrix} \begin{bmatrix} \tilde{B}_1 \\ \tilde{B}_2 \\ \vdots \\ \tilde{B}_\tau \end{bmatrix} =$$

$$= \begin{bmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_{\tau-1} & A_{\tau-2} & \cdots & A_0 \end{bmatrix} \begin{bmatrix} B_0 & 0 & \cdots & 0 \\ B_1 & B_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-2} & B_{\tau-3} & \cdots & B_0 \end{bmatrix} \begin{bmatrix} B_0 & 0 & \cdots & 0 \\ B_1 & B_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-2} & B_{\tau-3} & \cdots & B_0 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{B}_1 \\ \tilde{B}_2 \\ \vdots \\ \tilde{B}_\tau \end{bmatrix} =$$

$$= \begin{bmatrix} C_0 & 0 & \cdots & 0 \\ C_1 & C_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{\tau-1} & C_{\tau-2} & \cdots & C_0 \end{bmatrix} \begin{bmatrix} B_0 & 0 & \cdots & 0 \\ B_1 & B_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-2} & B_{\tau-3} & \cdots & B_0 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{B}_1 \\ \tilde{B}_2 \\ \vdots \\ \tilde{B}_\tau \end{bmatrix}.$$

Therefore, one solution of the system is given by:

$$\begin{bmatrix} \tilde{B}'_1 \\ \tilde{B}'_2 \\ \vdots \\ \tilde{B}'_{\tau-1} \end{bmatrix} = \begin{bmatrix} B_0 & 0 & \cdots & 0 \\ B_1 & B_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-2} & B_{\tau-3} & \cdots & B_0 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{B}_1 \\ \tilde{B}_2 \\ \vdots \\ \tilde{B}_{\tau-1} \end{bmatrix}.$$

After finding the sets of matrices $\mathcal{A}'$, $\mathcal{B}'$ and $\tilde{\mathcal{B}}'$, one can easily break the FAPKC by constructing an inverse transducer of the public key transducer M.

Example 5.3.4. *Let $M = \langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{12}, \delta, \lambda \rangle$ be the public key transducer with input memory 4 and invertible with delay 2 presented in the previous section (*Example *5.2.1) and defined by*

$$M : z_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_t$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

Since M is invertible with delay 2, one has to extend its memory to 5, because M_0' has to have input memory of order 2 and M_1' input memory of order 3. Therefore, consider M defined by

$$M : z_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+5} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1}$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4}x_{t+3} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}x_{t+2} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1}, \text{ for } t \geq 0.$$

It is easy to construct M_1' from A_0', A_1', A_2' since $A_j' = C_j$, for $j = 0, 1, 2$.

$$M_1' : z_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_{t+2} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_{t+1} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y_t, \text{ for } t \geq 0.$$

To construct M_0', one needs to find the matrices $B_0', B_1', B_2', \tilde{B}_1', \tilde{B}_2'$. The matrix B_0' is the identity matrix, $B_1', B_2' = 0$, and $\tilde{B}_1', \tilde{B}_2'$ are obtained by solving the following linear system:

$$\begin{bmatrix} C_0 & 0 \\ C_1 & C_0 \end{bmatrix} \begin{bmatrix} \tilde{B}_1' \\ \tilde{B}_2' \end{bmatrix} = \begin{bmatrix} \tilde{C}_1 \\ \tilde{C}_2 \end{bmatrix}.$$

One solution of the system is $\tilde{B}_1' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ and $\tilde{B}_2' = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$. Thus, M_0' is the transducer with input memory of order 3 defined by

$$M_0' : y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

The compound transducer $M' = M'_1 \circ M'_0$ *is given by:*

$$M' : z'_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+5} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t+3} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4}x_{t+3}$$
$$+ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1}x_t, \text{ for } t \geq 0.$$

Notice that the transducers M *and* M' *have the same coefficient matrices for the* 3 *most recent inputs, i.e.,* $z'_t = z_t + g(x_{t+2}, x_{t+1}, x_t)$*, for* $t \geq 0$*, where*

$$g(x_{t+2}, x_{t+1}, x_t) = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1}x_t.$$

This means that one is in conditions to apply Theorem *5.3.1, that allows to invert* M *from an inverse transducer of* M'*. To invert* M' *one just has to compute* $M'^{-1} = M'^{-1}_0 \circ M'^{-1}_1$*. Let* $M'^{-1} = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \mathbb{F}_2^{21}, \delta'^{-1}, \lambda'^{-1} \right\rangle$ *be an inverse transducer of* M' *defined by*

$$M'^{-1} : x_{t+5} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} z'_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} z'_t + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+4} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}$$
$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x_{t+4}x_{t+3} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+3}x_{t+2} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{t+2}x_{t+1} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x_{t+1}x_t.$$

Recall that, from Example *5.2.1, the initial state of the transducer* M *is* $s =< \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} >$.

Since M' *was extended, its initial state is* $s' =< 0, x_0, x_1, x_2, x_3 >= < \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} >$.

Given the ciphertext $z_0z_1z_2z_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$*, and because* M' *is invertible with delay* 2*,*

the initial state of M'^{-1} *is* $s'^{-1} =< z'_0, z'_1, 0, x_0, x_1, x_2, x_3 >$*, where*

$$z'_0 = z_0 + g(x_1, x_0, 0) = \begin{bmatrix}1\\0\\0\end{bmatrix} + g\left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}\right) = \begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix},$$

$$z'_1 = z_1 + g(x_2, x_1, x_0) = \begin{bmatrix}0\\1\\0\end{bmatrix} + g\left(\begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix}.$$

To decrypt z_2z_3*, one has to proceed as follows: compute* z'_2 *from* z_2*, recover* x_4 *using the transducer* M'^{-1}*, and repeat the procedure to* z_3*.*

$$z'_2 = z_2 + g(x_3, x_2, x_1) = \begin{bmatrix}1\\0\\1\end{bmatrix} + g\left(\begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}\right) = \begin{bmatrix}1\\0\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix}.$$

$$x_4 = \lambda'^{-1}(< z'_0, z'_1, 0, x_0, x_1, x_2, x_3, >, z'_2) = \begin{bmatrix}1\\0\\0\end{bmatrix}.$$

$$z'_3 = z_3 + g(x_4, x_3, x_2) = \begin{bmatrix}1\\1\\0\end{bmatrix} + g\left(\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

$$x_5 = \lambda'^{-1}(< z'_1, z'_2, 0, x_1, x_2, x_3, x_4 >, z'_3) = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

The plaintext recover, i.e., $x_4x_5 = \begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\0\\0\end{bmatrix}$*, is the correct one (as can be confirmed with* Example *5.2.1).*

Chapter 6

Conclusion

In this work, we presented concepts and known results on general finite transducers as well as on finite transducers with memory, linear and quasi-linear ones. We simplified the language used by Tao and illustrated the concepts with a wide variety of examples. Furthermore, we extended the definition of quasi-linear finite transducers with memory, presented by Tao without any justification to the fact that the most recent inputs only appear in the linear part.

We formalized Tao's method of checking injectivity of linear finite transducers with memory, with an algorithm that simultaneously obtains an inverse transducer. Also, we presented a necessary and sufficient condition to invertibility of linear finite transducers with memory. Considering the difficulty of this procedure, we illustrated all phases through a simple example. The same was done to quasi-linear finite transducers, and all results were extended to our new definition.

We presented a new formalization of the two different ways to compose finite transducers described by Tao, and presented results regarding the order of the memory and the injectivity delay of compound transducers with respect to its factors. We given a general description of all FAPKCs by means of a general scheme, the only one that we can understand through the papers we had access. For this base scheme, we presented a key generation procedure as well as the encryption and decryption processes. Lastly, we presented the Bao-Igarashi attack to FAPKC. This attack was never illustrated through an example, and the related papers lack complete proofs and explanations. We formalized and extended this attack to all cases that involve invertible linear transducers with delay 0, since the original attack only works for special cases. We also illustrated

this attack through an example.

For future work, it will be important to understand the other variants of FAPKC, particularly, how the key generation could be done. After that, one should consider the possibility of extending the Bao-Igarashi attack to other FAPKC schemes. It is conceivable that this attack can be modified to factor compound transducers obtained from transducers with non-zero delay. Although many papers refer that the FAPKC schemes after FAPKC2 are not vulnerable to this attack, no evidence has been provided so far. Another fundamental direction of research is the study of general non-linear finite transducers and their invertibility.

Appendix A

Number of Verifications Needed to Test Invertibility of Transducers

Let $M = \langle \mathcal{X}, \mathcal{Y}, \mathcal{X}^h \times \mathcal{Y}^k, \delta, \lambda \rangle$ be a finite transducer with memory of order (h, k). Considering the special structure of finite transducers with memory, it is plausible that the number of checks required to see if M is ω-injective is lower than $\frac{|S|(|S|-1)}{2}$, where $S = \mathcal{X}^h \times \mathcal{Y}^k$. In fact, we suspect that M is ω-injective if and only if there exists a non-negative integer $\tau \leq h \dim(\mathcal{X})$ such that M is injective with delay τ.

First, we start by doing some practical tests with linear finite transducer with only input memory, since the problem of checking injectivity can be reduced to these transducers. In the first tests, we check τ-injectivity for all possible linear finite transducers with input memory of order h, for $h = 1, 2, 3$, over $\mathbb{F}_2$, i.e., all possible transducers M such that $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^2, \left(\mathbb{F}_2^2\right)^h, \delta, \lambda \right\rangle$. The results obtained are summarized in the next table.

	number of τ-injective transducers							not ω-injective	total	$\frac{\lvert S\rvert(\lvert S\rvert-1)}{2}$
	$\tau=0$	$\tau=1$	$\tau=2$	$\tau=3$	$\tau=4$	$\tau=5$	$\tau=6$			
$h=1$	96	78	18	-	-	-	-	64	256	6
$h=2$	1536	1248	654	234	72	-	-	352	4096	120
$h=3$	24576	19968	10464	5262	2250	936	288	1792	65536	2016

Table A.1: Injectivity of transducers $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^2, \left(\mathbb{F}_2^2\right)^h, \delta, \lambda \right\rangle$, for $h = 1, 2, 3$

From the results presented, one can see that the linear finite transducers of the form $M = \left\langle \mathbb{F}_2^2, \mathbb{F}_2^2, \left(\mathbb{F}_2^2\right)^h, \delta, \lambda \right\rangle$ with input memory h, for $h = 1, 2, 3$, are ω-injective if and only if they are τ-injective for $\tau \leq h \dim(\mathbb{F}_2^2) = 2h$.

Let $\ell \in \mathbb{N}$. We are able to construct a generic linear finite transducer $M = \left\langle \mathbb{F}_2^\ell, \mathbb{F}_2^\ell, \left(\mathbb{F}_2^\ell\right)^h, \delta, \lambda \right\rangle$, with input memory of order $h \in \mathbb{N}$, that is invertible with delay $\tau = h\ell$:

$$M : y_t = \begin{bmatrix} Id_{\ell-1} & 0_{(\ell-1)\times 1} \\ 0_{1\times(\ell-1)} & 0 \end{bmatrix} x_{t+h} + \begin{bmatrix} 0_{(\ell-1)\times 1} & Id_{\ell-1} \\ 1 & 0_{1\times(\ell-1)} \end{bmatrix} x_t, \text{ for } t \geq 0.$$

Example A.1. *Let $M = \left\langle \mathbb{F}_2^4, \mathbb{F}_2^4, \left(\mathbb{F}_2^4\right)^2, \delta, \lambda \right\rangle$ be the linear finite transducer with memory of order 2 defined by:*

$$M : y_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x_{t+2} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x_{t+1} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} x_t, \quad \text{for } t \geq 0.$$

M is invertible with delay $\tau = 2 \times 4 = 8$.

To understand the construction of the generic example, one has to notice that the coefficient matrix of x_{t+h} is in reduced form and only has one null row. The information missing in this matrix is related to the last component of x_{t+h}. If one applies a R_b transformation, the last row is shifted one matrix to the left. It is necessary h R_b transformations to enter a non-null row in the coefficient matrix of x_{t+h}. The first non-null row arriving at this matrix is equal to the first row, therefore, one has to apply a R_a transformation, adding the first row to the last one. After this transformation, the coefficient matrix of x_{t+h} continues equal to the inicial one, but in the last row of the coefficient matrix of x_t appears the first row. After another h R_b transformations, this row enters in the coefficient matrix of x_{t+h} but it is cancelled by its second row in the R_a transformation. This process repeated ℓ times, until the row $[00\ldots01]$ enters in the first matrix, which now has full rank. Therefore, the transducer is invertible with delay $h\ell$.

The question now is if it is possible to have other matrices instead of the null ones that increase the delay. In order to test this hypothesis, we consider the linear finite transducers $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \left(\mathbb{F}_2^3\right)^h, \delta, \lambda \right\rangle$, for $h = 2, 3$, and we replace the null matrices by all possible matrices with dimension 3 over $\mathbb{F}_2$. Then, we check the transducers injectivity. The results obtained are in the next table.

	number of τ-injective transducers		not	total	$\frac{\lvert S\rvert(\lvert S\rvert-1)}{2}$
	$\tau \leq 3h$	$\tau > 3h$	ω-injective		
$h = 2$	512	0	0	512	2016
$h = 3$	262144	0	0	262144	130816

Table A.2: Injectivity of a subset of transducers $M = \left\langle \mathbb{F}_2^3, \mathbb{F}_2^3, \left(\mathbb{F}_2^3\right)^h, \delta, \lambda \right\rangle$, for $h = 2, 3$

From the previous table, we have that none of the transducers generated has delay greater than $3h$.

Although the tests presented may not be statistically relevant, we don't find any transducer that refutes our claim. For future work, it will be important to prove this result.

References

[AMR12] Ivone Amorim, António Machiavelo, and Rogério Reis. Formal power series and the invertibility of finite linear transducers. In *NCMA*, pages 33–48, 2012.

[AMR14a] Ivone Amorim, António Machiavelo, and Rogério Reis. Counting equivalent linear finite transducers using a canonical form. In *International Conference on Implementation and Application of Automata*, pages 70–83. Springer, 2014.

[AMR14b] Ivone Amorim, António Machiavelo, and Rogério Reis. On the invertibility of finite linear transducers. *RAIRO-Theoretical Informatics and Applications*, 48(1):107–125, 2014.

[AMR14c] Ivone Amorim, António Machiavelo, and Rogério Reis. Statistical study on the number of injective linear finite transducers. *arXiv preprint arXiv:1407.0169*, 2014.

[BI95] Feng Bao and Yoshihide Igarashi. *Break Finite Automata Public Key Cryptosystem*, pages 147–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

[dCA16] Ivone de Fátima da Cruz Amorim. Linear finite transducers towards a public key cryptographic system. 2016.

[Dif88] Whitfield Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76(5):560–577, 1988.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[McC71] Neal H. McCoy. *Introduction to modern algebra*. Allyn and Bacon, 1971.

[Mil85] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 417–426. Springer, 1985.

[Tao95] Renji Tao. On ra rb transformation and inversion of compound finite automata. Technical report, Technical Report No. ISCAS-LCS-95-10, Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 1995.

[Tao09] Renji Tao. *Finite automata and application to cryptography*. Springer, 2009.

[TC86] Renji Tao and Shihua Chen. Two varieties of finite automaton public key cryptosystem and digital signatures. *Journal of Computer Science and Technology*, 1(1):9–18, Mar 1986.

[TC97] Renji Tao and Shihua Chen. A variant of the public key cryptosystem fapkc3. *Journal of Network and Computer Applications*, 20(3):283 – 303, 1997.

[TC99] Renji Tao and Shihua Chen. The generalization of public key cryptosystem fapkc4. *Chinese Science Bulletin*, 44(9):784–790, May 1999.

[TCC97] Renji Tao, Shihua Chen, and Xuemei Chen. Fapkc3: A new finite automaton public key cryptosystem. *Journal of Computer Science and Technology*, 12(4):289, Jul 1997.

[Val93] Robert J Valenza. *Linear algebra: an introduction to abstract mathematics*. Springer Science & Business Media, 1993.